Nortel Networks Symposium Agent

Programmer's Guide

Product release: 2.3

Document release: Standard 2.0

Date: January 2001

NORTEL NETWORKS™

How the world shares ideas.

Nortel Networks Symposium Agent

Programmer's Guide

Product release:	2.3
Document release:	Standard 2.0
Date:	January 2001

Copyright © 2001 Nortel Networks, All Rights Reserved.

Information is subject to change without notice. Nortel Networks reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

The process of transmitting data and call messaging between the Meridian 1 and the Symposium Agent is proprietary to Nortel Networks. Any other use of the data and the transmission process is a violation of the user license unless specifically authorized in writing by Nortel Networks prior to such use. Violations of the license by alternative usage of any portion of this process or the related hardware constitutes grounds for an immediate termination of the license and Nortel Networks reserves the right to seek all allowable remedies for such breach.

DMS, MERIDIAN 1, MERIDIAN LINK, MSL-100, NORTEL NETWORKS, NORTEL NETWORKS HOW THE WORLD SHARES IDEAS, and SYMPOSIUM are trademarks of Nortel Networks. INTERNET EXPLORER, WINDOWS, and WINDOWS NT are trademarks of Microsoft Corporation.

Your responsibility for your system's security

You are responsible for the security of your system. Nortel Networks does not warrant that this product is immune from or prevents unauthorized use of common-carrier telecommunication services or facilities accessed through or connected to it. Nortel Networks is not responsible for any charges that result from such unauthorized use. Product administration to prevent unauthorized use is your responsibility, and your system administrator should read all documents provided with this product to fully understand the features available that may reduce your risk of incurring charges.

NORTEL NETWORKS™

How the world shares ideas.

Publication history

July 2000	This is the Draft 0.01 release of the *Nortel Networks Symposium Agent Release 2.3 Programmer's Guide*.
November 2000	This is the Draft 0.02 release of the *Nortel Networks Symposium Agent Release 2.3 Programmer's Guide* incorporating recommended changes from Technology.
December 2000	This is the Standard 1.0 release of the *Nortel Networks Symposium Agent Release 2.3 Programmer's Guide* incorporating recommended changes from Technology.
January 2001	This is the Standard 2.0 release of the *Nortel Networks Symposium Agent Release 2.3 Programmer's Guide* incorporating recommended changes from Technology.

Contents

Getting started 1

About this guide....2
Overview....3
Minimum system requirements....5
Symposium Agent data flow....6

Setting up triggers and call rules 9

Setting up a keystroke call trigger....10
Setting up a DDE trigger....15
Setting up an OLE trigger....18
Setting up a call rule....28
Setting up a call rules group....31

Using Agent Compass 33

Symposium Agent data flow in agentCompass....34
Query strings....35
Web server setup....37
Client setup....38
Setting up an agentExplorer call trigger....39
Setting up and enabling agentCompass call rules....41
Setting up telephony-enabled tabs for agents....42
Testing the call rule....44

Getting Started with Symposium Agent Reporting 45

Overview....46
Configuring the database access....48
Defining an ODBC System DSN....52
Database Entity Relationship Diagram....55
Data Dictionary....57
Sample Report using Crystal Reports....70

Outbound dialing using Symposium Agent 75

Dial feature....76

Active Server Page examples 78

Introduction....79
An Active Server Page sample (NNWEB.ASP)....80
Sample source for switchboard.asp....82
Sample source for flowers.asp....83

Sample source for pizza.asp 84
Sample source for unknown.asp 85

Index **87**

Chapter 1

Getting started

In this chapter

About this guide	2
Overview	3
Minimum system requirements	5
Symposium Agent data flow	6

About this guide

Introduction

The *Symposium Agent Release 2.3 Programmer's Guide* describes how to create OLE, DDE, and keystroke macro triggers. This guide also demonstrates the use of the agentCompass module in Symposium Agent Release 2.3.

Distribution of this guide

This document is available only in electronic Portable Document Format (PDF) on the CD-ROM that is supplied with the Symposium Agent Release 2.3 software.

View this file using Acrobat Reader, or print it in whole or in part for individual use.

Who should use this guide

Information in this document is intended for developers writing web applications that use Symposium Agent Release 2.3.

Skills you need

To use this guide effectively, you must have experience with basic programming, telephony products, and Microsoft Windows. You need to be able to write triggers and call rules for Symposium Agent, and you should be familiar with Visual Basic or another language that supports OLE Automation technology. In addition, if you plan to use the agentCompass feature, you should have experience with server-side web programming.

For further information

For further information on using Call Rules, refer to the Call Rules topic in the Symposium Agent online Help.

For further information about Symposium Agent, refer to the *Symposium Agent Release 2.3 Installation Guide*.

For additional information or help using this guide, call the Nortel Networks Developer Support Hotline at (800) NT-4CTI-0.

For information about other Nortel Networks products, call (800) 4-NORTEL (466-7835). Outside the United States and Canada, contact your Nortel Support or Sales representative.

Overview

Symposium Agent Release 2.3 provides several methods for passing telephony information to various applications, including web applications. This information can be passed to applications such as Microsoft Outlook, Microsoft Excel, or a custom application using OLE Automation, Dynamic Data Exchange (DDE), or keystroke macros. Symposium Agent also provides the agentCompass feature that allows a user to pass telephony information to a web application.

In Symposium Agent Release 2.3, the AgentCompass feature is provided as a built-in tool that allows you to telephony-enable any web page or web application. The Symposium Agent Release 2.3 client has built-in web browsers called agentTabs that use Internet Explorer 4.0 browser technology. The agentCompass feature is available only to the Symposium Agent browser tabs and provides a means for passing telephony data and additional information to any web page that can reside in Internet Explorer 4.0.

AgentCompass passes telephony information, such as caller telephone number or data collected by an Integrated Voice Response (IVR) unit, to a web server. This data can be processed by any current web technology. For example, agentCompass works with web pages using Active Server Pages (ASP), PERL, JavaScript, and VB Script.

AgentCompass uses Symposium Agent's support for OLE call rules to communicate with the Symposium Agent agentTabs. An administrator uses the Symposium Agent Call Rules Explorer to create call rules that define when telephony information is passed to the agentTabs (URL). As with any call rule, the administrator can designate when the call rule executes. Call rules can be set up to execute when an incoming call is ringing, answered, or released, and when an outgoing call is answered or released. When a call rule is executed, telephony information can be passed to all the agentTabs.

The following screen capture shows an example of an agentTab in Symposium Agent 2.3.

Minimum system requirements

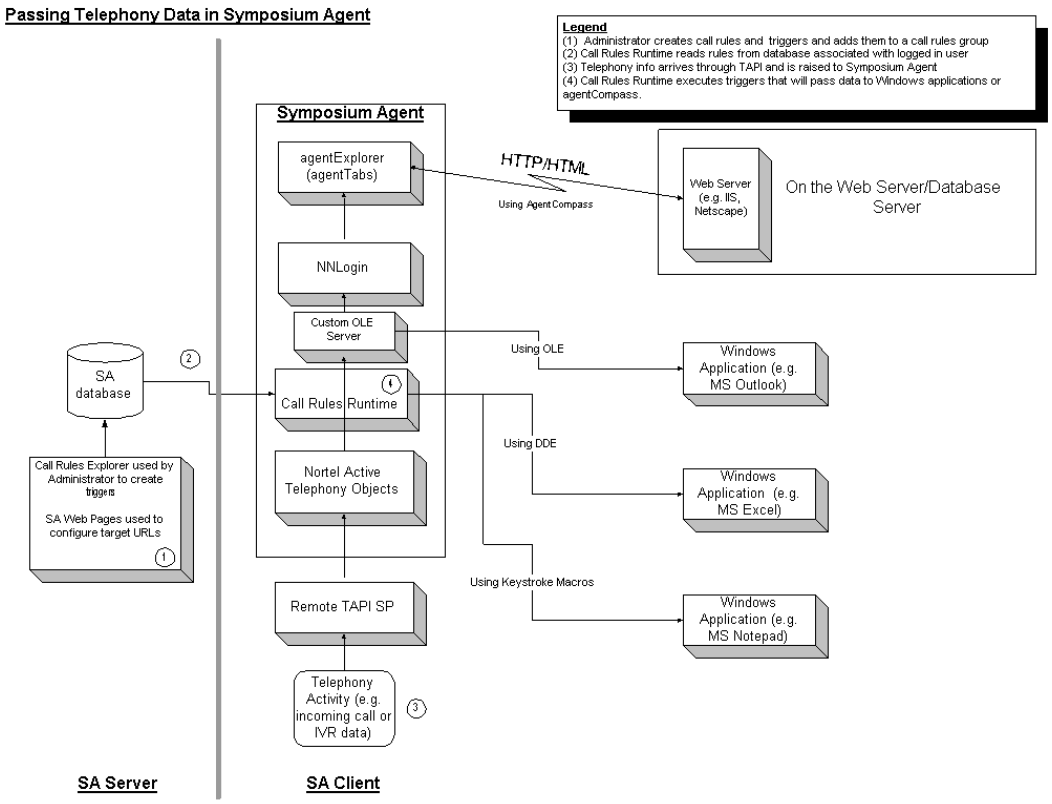
- Symposium Agent 2.3 Server for Windows NT 4.0 with a minimum of Service Pack 4 installed
- Symposium Agent 2.3 Client for Windows 95, Windows 98, or Windows NT with a minimum of Service Pack 4 installed
- Internet Explorer 4.0 or 5.0 (a requirement for the Symposium Agent Release 2.3 client and server installation)

For the complete specifications required for your configuration, refer to the *Symposium Agent Release 2.3 Planning and Engineering Guide*.

.

Symposium Agent data flow

The following diagram shows how data is passed by Symposium Agent.


Passing Telephony Data in Symposium Agent
Legend
(1) Administrator creates call rules and triggers and adds them to a call rules group
(2) Call Rules Runtime reads rules from database associated with logged in user
(3) Telephony info arrives through TAPI and is raised to Symposium Agent
(4) Call Rules Runtime executes triggers that will pass data to Windows applications or agentCompass.
Symposium Agent
agentExplorer (agentTabs)
HTTP/HTML
Using AgentCompass
Web Server (e.g. IIS, Netscape)
On the Web Server/Database Server
NNLogin
Custom OLE Server
Using OLE
Windows Application (e.g. MS Outlook)
SA database
Call Rules Runtime
Using DDE
Windows Application (e.g. MS Excel)
Call Rules Explorer used by Administrator to create triggers
SA Web Pages used to configure target URLs
Nortel Active Telephony Objects
Using Keystroke Macros
Windows Application (e.g. MS Notepad)
Remote TAPI SP
Telephony Activity (e.g. incoming call or IVR data)
SA Server
SA Client

During initialization, Call Rules Runtime retrieves the call rules and triggers from the Symposium Agent database for the currently logged in user. The call is passed to the TAPI interface, which raises the telephony data to the Call Rules Runtime. Rules Runtime initiates the correct trigger based on call criteria such as call state (for example, on Answer). OLE triggers raise call data to windows applications that support OLE automation using a custom OLE Automation Server. DDE triggers and keystroke macros are executed from the call rules runtime component. The agentCompass feature uses an OLE DLL to communicate with NNLogin. OLE, DDE, and keystroke macro triggers are discussed in further detail in the following chapter.

The following data is passed when a call rule is executed.

Call rule data

CallerNumber	Number of party who placed the call.
CalledNumber	Number of party or DNIS receiving the call.
CallData	Attached data such IVR data.
CallID	The callID is the unique identifier for a call in a networked environment.
State	This is the call state (Connected, Idle, Offering).
Extension	The extension on which the call arrived.
LineName	The line name on which the call arrived.
TrunkInfo	TrunkInfo consists of two numbers: the trunk group and member for calls outside the local switch. This information might be used by voice recording equipment which sits on a trunk and records calls based on trunk group and member numbers.
CallerName	This is the caller name. Since this does not always show up, it was placed at the end of the query string.
CalledName	This is the called name. Since this does not always show up, it was placed at the end of the query string.

The Symposium TAPI Service Provider does not support CallerName and CalledName for the Meridian 1. CallerName and CalledName are supported by the ICM TAPI Driver when used with the DMS or MSL-100.

Additional data is available when using the agentCompass feature. This information is described in the section *Symposium Agent data flow in agentCompass* on page 34.

Chapter 2

Setting up triggers and call rules

In this chapter

Setting up a keystroke call trigger	10
Setting up a DDE trigger	15
Setting up an OLE trigger	18
Setting up a call rule	28
Setting up a call rules group	31

Setting up a keystroke call trigger

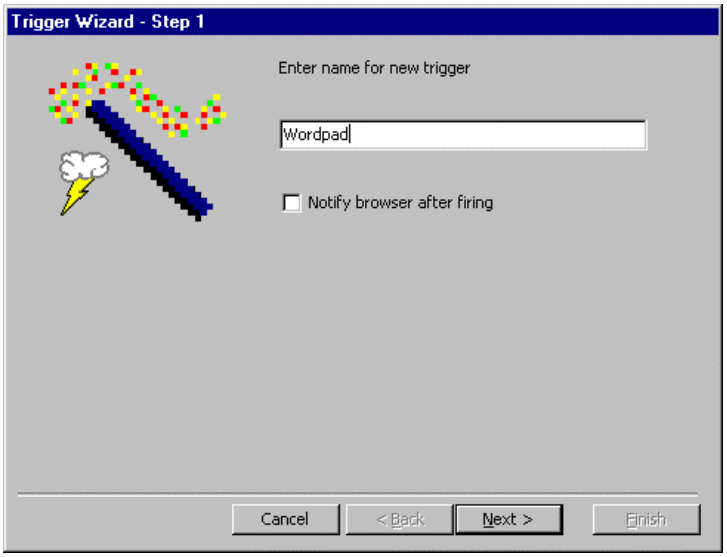
Applications that do not support OLE Automation or the use of DDE can still receive telephony information by using keystroke macros. The following example uses Microsoft Wordpad.

To set up a keystroke trigger

1 Log in to Symposium Agent Administration and open Call Rules Explorer.

2 Click **Trigger Wizard** on the *Tools* menu. (You can also open the Trigger Wizard by right-clicking the Triggers icon and selecting New Trigger Wizard.)

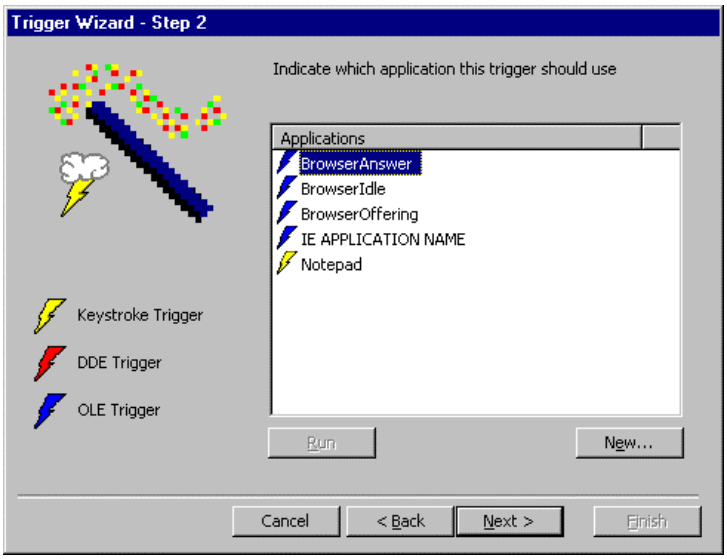
The *Trigger Wizard – Step 1* dialog box appears.



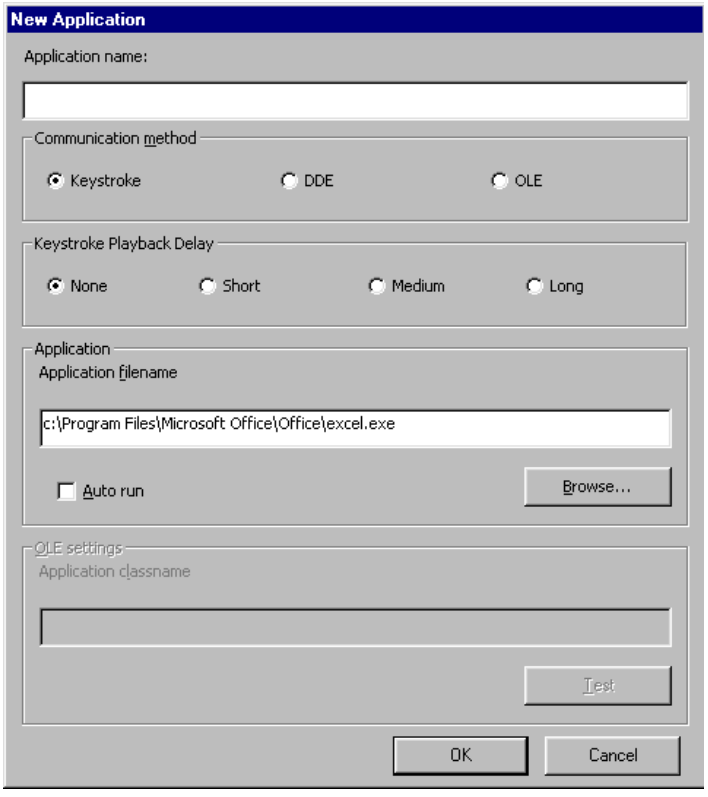
3 Type a name for this trigger.

4 Check the **Notify browser after firing** check box.

5 Click **Next** to move to the *Trigger Wizard Step 2* dialog box.



6 Since the trigger is not in the list, click **New** to move to the *New Application* dialog box.

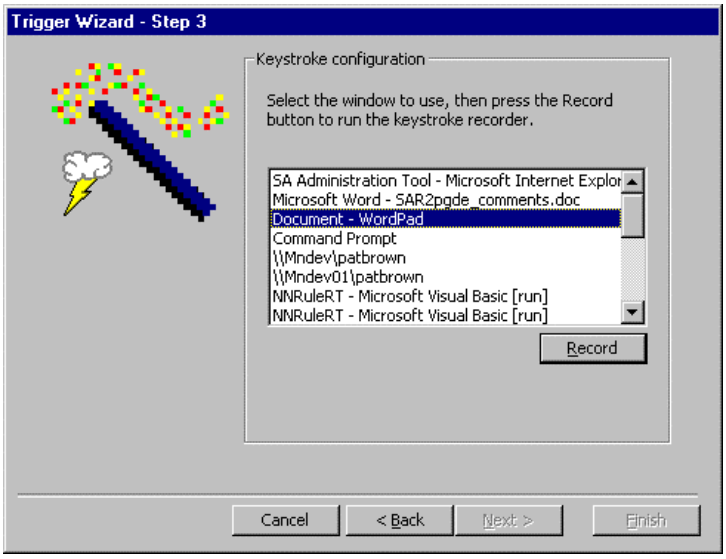


7 Type the Application name. Make sure this name exactly matches the window title that will appear when it screen pops at the agent's PC. You may have to maximize the application window to check its title.

8 Type the Application filename. This name has an .exe extension.

9 Open the application you just named if you have not already done so.

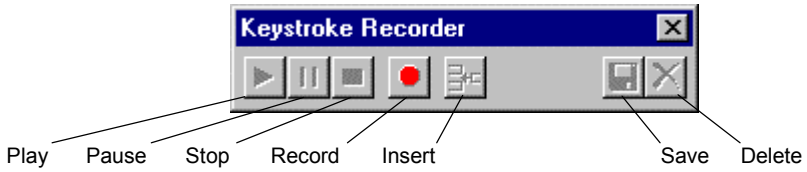
10 Click **OK** to move to the *Trigger Wizard – Step 3* dialog box.



11 Select the application from this list.

12 Click **Record** to open the Keystroke Recorder toolbar.

The Keystroke Recorder toolbar appears in the lower right corner of your screen to avoid conflict with other windows. It is used to record keystrokes and to send telephony data to the application.



13 Click the red **Record** button and begin recording the keystroke sequence required to open the application or a file within the application.

Note: You must use keystrokes to record all the actions you want to occur on playback. Mouse movements will not be recorded. For example, enter the **Alt-F** keystroke sequence instead of using the mouse to click **File** on the menu bar.

14 If telephony items are required, insert them as follows:

a. Click the **Pause** button, then click the **Insert** button. Do not click the black Stop button as this will terminate the process. The Insert Data Wizard appears.

b. Select the items to insert, then click **Next**. You can insert the following data:

Caller number	This is the number of the party who placed the call.
Caller name	This is the caller name (DMS/MSL-100 only).
Called number	This is the number of the party receiving the call.
Called name	This is the called name (DMS/MSL-100 only).
Call data	This is attached data, such as IVR data.

Note: If you select Call data with an outbound call rule, the trigger executes only

when the call is answered, not on dial tone.

c. Select the **type of data** you wish to insert. The type of data is a pre-existing format you want to apply to the caller or called number.

d. Click **Next**.

e. Click the **Record** button again to continue recording keystrokes.

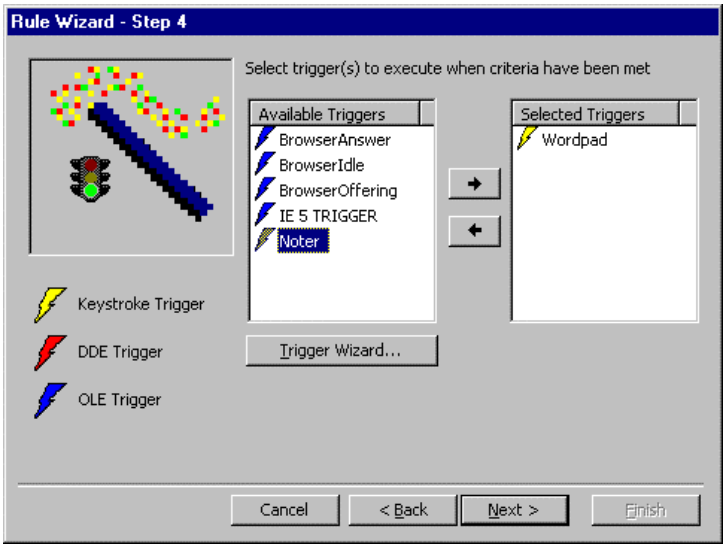
15 When you have finished recording, click the **Stop** button.

16 At this point, you can play, save, or delete the recorded keystrokes.

17 Test your keystroke macro by clicking the **Play** button. If you specified that call data should be added to the keystroke sequence, the *Sample Data Values* dialog box appears. This dialog box allows you to enter test values for any call data that will be played back during this test process. Enter the call data and click the **Execute** button.

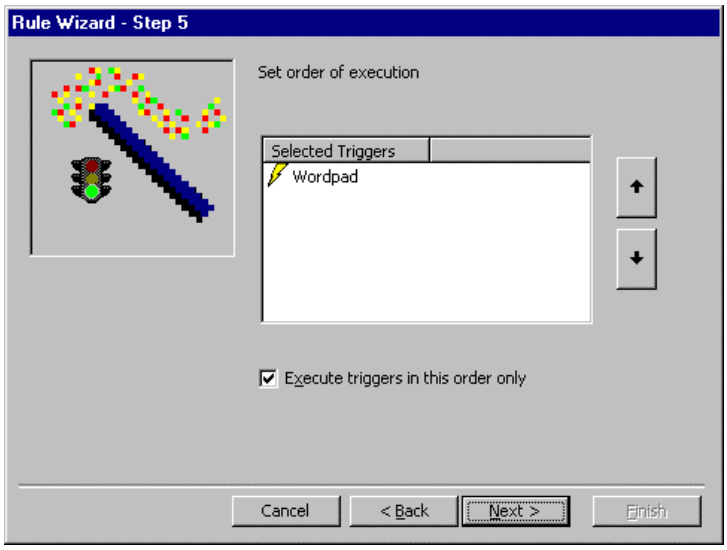
18 To save your keystroke macro, click the **Save** button on the Keystroke Recorder toolbar. This saves the keystroke sequence and ends the trigger creation process. The *Trigger Wizard – Finished!* dialog box appears.

19 Click **Finish** to move to the *Rule Wizard – Step 4* dialog box.

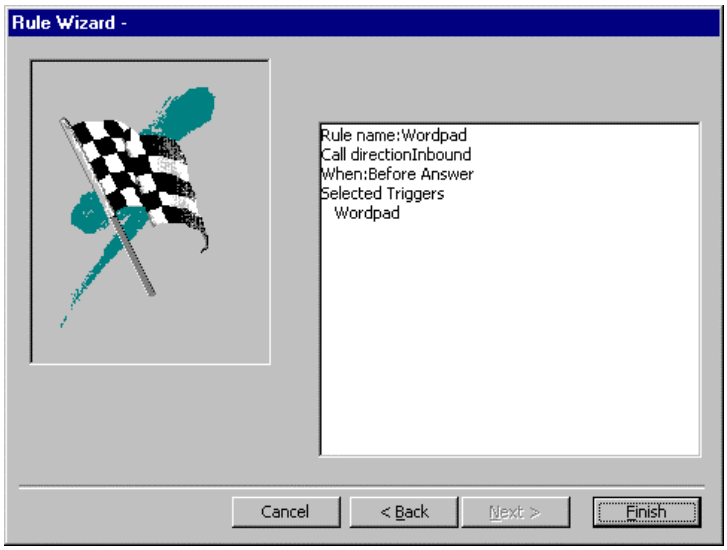


20 From the Available Triggers list, highlight the trigger you want to execute.

21 Click the right arrow to move the highlighted trigger to the Selected Triggers list. When you have selected all the triggers required for this call rule, click **Next** to move to the *Rule Wizard – Step 5* dialog box.



22 The dialog box indicates the trigger firing order of this rule. Change the order if required, using the up and down arrows. When you are finished, click **Next** to move to the *Rule Wizard* – dialog box.



23 Check that the summary of the call rule configuration is correct.

24 Click **Finish**. The Call Rules Explorer window opens displaying the new call rule.

25 Select **File/Close** to exit Call Rules Explorer.

Setting up a DDE trigger

Dynamic data exchange (DDE) is the ability to exchange information between applications or pass information to an application. These exchanges, called conversations, are the messages sent to and from Symposium Agent and other applications.

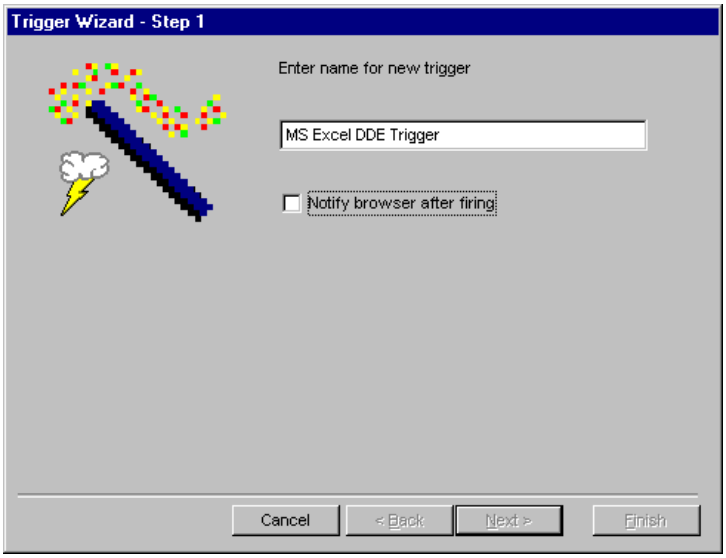
The following example illustrates the DDE method of communication. In this example, Microsoft Excel is used as a DDE application. Using DDE Poke, this test places the caller number into a spreadsheet at location row 1, column 1.

To set up a DDE trigger

1 Log in to Symposium Agent Administration and open Call Rules Explorer.

2 Click **Trigger Wizard** on the *Tools* menu. (You can also open the Trigger Wizard by right-clicking the Triggers icon and selecting New Trigger Wizard.)

The *Trigger Wizard – Step 1* dialog box appears.

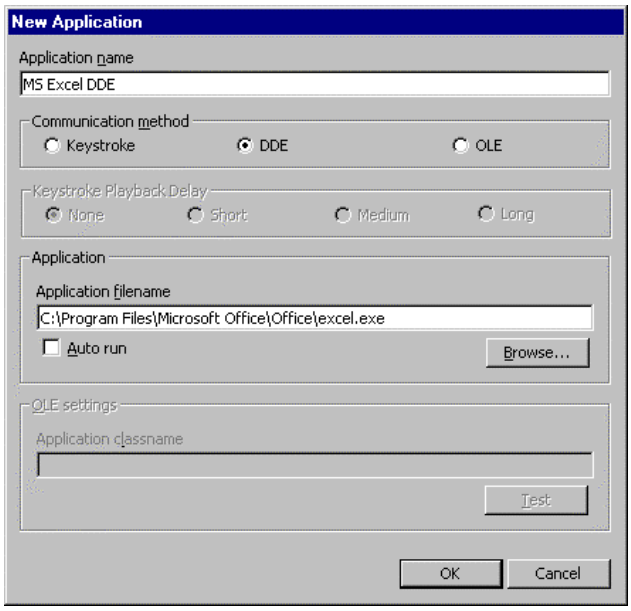


3 Type a name for this trigger.

4 Uncheck the **Notify browser after firing** check box.

5 Click **Next** to move to the *Trigger Wizard Step 2* dialog box.

6 Click **New** to move to the *New Application* dialog box.



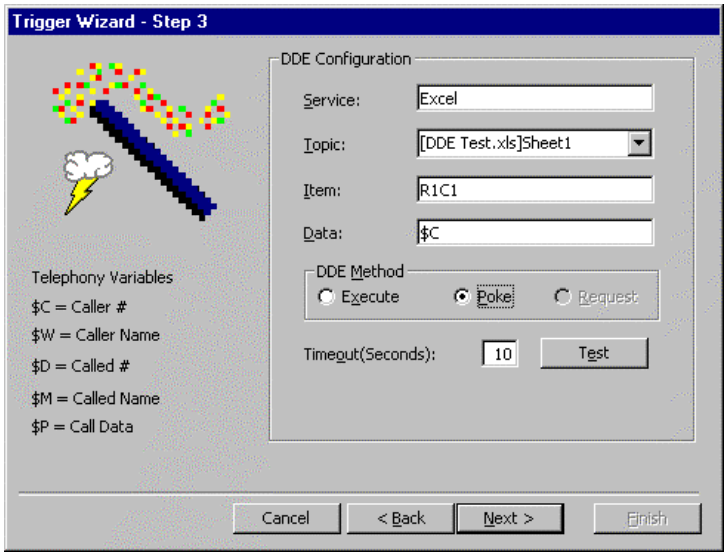
7 Type the Application name, Communication method, and Application name. An example is shown below.

Example of application information

Application name	MS Excel DDE
Communication method	DDE
Application name	Excel.exe

8 Click **OK** to return to the *New Application* dialog box.

9 Click **Next** to move to the *Trigger Wizard – Step 3* dialog box and configure the DDE properties.



Service: The name of the DDE application. In this case, Service is Excel.

Topic: The Excel spreadsheet (file name and sheet name, e.g. DDE Test.xls).

Item: The location within the spreadsheet—row 1, column 1.

Data: The data sent to Excel. In this case, the Call Rules Runtime will send the caller's phone number ($C).

$C	Caller #
$W	Caller Name
$D	Called #
$M	Called Name
$P	Call Date

DDE Method: Two methods are supported: Execute and Poke. Execute is used to call a procedure in the DDE application. Poke is used to "push" data to that application. Since this test "pushes" data to row 1, column 1 of the spreadsheet DDE Test.xls, Poke is used.

10 Click **OK** to return to the New Application dialog box. Highlight the application MS Excel DDE.

11 Click **Next** to move to the *Trigger Wizard – Finished!* dialog box

12 The dialog box summarizes the information for the trigger. Click **Finish** to open the *Call Rules Explorer* window and begin the next procedure.

Setting up an OLE trigger

OLE triggers communicate with windows applications that support OLE automation (Object Linking and Embedding). Before creating an OLE trigger, an OLE automation server must be created in order to communicate with the target application. The Call Rules Runtime component expects this server to support a specific interface. This interface is shown below.

OLE automation server properties and methods

The following code sample (written in Visual Basic) demonstrates the property and method declarations that implement the automation server interface. The five properties are set by Call Rules Runtime before invoking the method ExecuteTrigger. The properties reflect the telephony information of the current call.

```
‘*** ‘Private Properties ***

Private m_sCallerNumber As String
Private m_sCalledNumber As String
Private m_sCallData As String
Private m_sCallID As String
Private m_sUserData As String

‘*** Public Properties that must be supported by the OLE Automation Server ***

Public Property Let CallerNumber(asNumber As String)
    m_sCallerNumber = asNumber
End Property
Public Property Let CalledNumber(asNumber As String)
    m_sCalledNumber = asNumber
End Property
Public Property Let CallData(asData As String)
    m_sCallData = asData
End Property
Public Property Let CallID(asCallID As String)
    m_sCallID = asCallID
End Property
Public Property Let UserData(asUserData As String)
    m_ sUserData = asUserData
End Property

‘*** Public method that must be supported by the OLE Automation Server ***
‘*** This example uses Microsoft Outlook ***

Public Sub ExecuteTrigger()

    ‘*** Code to communicate with Target application ***

    Dim MyOutlook As Outlook.Application
    Dim MyContacts As Outlook.Items
    Dim MyItem As Object
```

```
    Dim sFilter As String
    Dim sNumber As String

    Select Case Len(m_sCallerNumber)
    Case 4
        sNumber = m_sCallerNumber
    Case 7
        sNumber = Format(m_sCallerNumber, "#######")
        sNumber = "6" & sNumber
    Case 10
        sNumber = Format(m_sCallerNumber, "(###) ###-####")
    Case Else
        sNumber = m_sCallerNumber
    End Select

    sFilter = "[Business Phone] = " & Chr$(34) & sNumber & Chr$(34)

    Set MyOutlook = GetObject(, "Outlook.Application.8")
    Set MyContacts = MyOutlook.GetNamespace("MAPI").GetDefaultFolder(olFolder
    Contacts).Items

    Set MyItem = MyContacts.Find(sFilter)

    Do Until (MyItem Is Nothing)
        MyItem.Display
        Set MyItem = MyContacts.FindNext
    Loop

    Set MyItem = Nothing
    Set MyContacts = Nothing
    Set MyOutlook = Nothing

End Sub
```

The Let Properties are used by Call Rules Runtime to pass telephony data to the application. The data passed to the application is configurable within the Call Rules Explorer. The ExecuteTrigger method is called by Call Rules Runtime after the criteria for the call rule have been met (calling number, called number, time of day, and so on) and the properties have been set. The code placed in ExecuteTrigger is up to the developer of the automation server. Typically, this code will invoke methods in the target application to perform database searches based on the supplied telephony information. The example code above presents a simple OLE automation server that accepts a caller's phone number and instructs Microsoft Outlook to search for that phone number in its contact database.

Note: This sample code was written in Visual Basic 5. Other languages that support OLE can be used with the call rules product.

There are times when an OLE trigger is written to expect call data (that is, IVR digits) or caller number, or both, but that data is not available for the call. For example, the customer may not have entered any data at an IVR prompt and "zeroed out" to speak to an agent, or has blocked his or her caller ID. In this case,

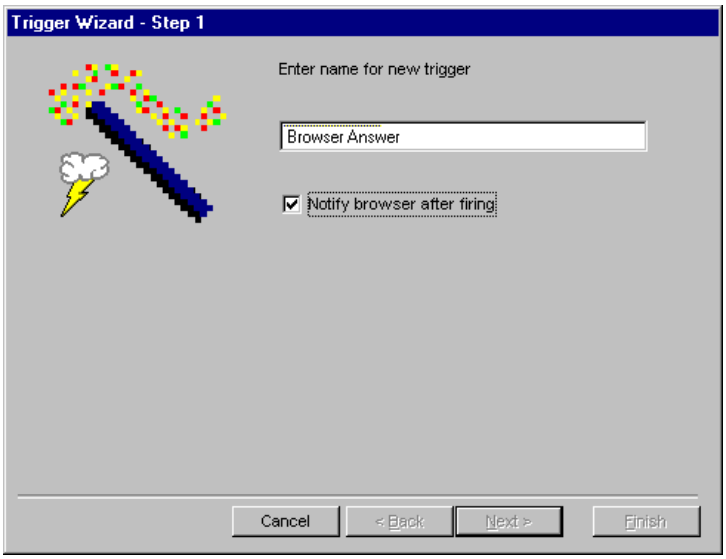
Symposium Agent realizes that this data will never be available and calls the ExecuteTrigger method with the unavailable properties set to Null strings. It is up to the OLE trigger code to realize that the properties are Null and take whatever action is deemed appropriate. In some cases, this might mean that the trigger does nothing. In other cases, the trigger might bring up a blank customer record for the agent. CallID is a unique identifier for the call. It is always passed to the OLE trigger before calling ExecuteTrigger. CallID is guaranteed to be unique for the life of the call.

To set up an OLE trigger

1 Log in to Symposium Agent Administration and open Call Rules Explorer.

2 Click **Trigger Wizard** on the *Tools* menu. (You can also open the Trigger Wizard by right-clicking the Triggers icon and selecting New Trigger Wizard.)

The *Trigger Wizard – Step 1* dialog box appears.

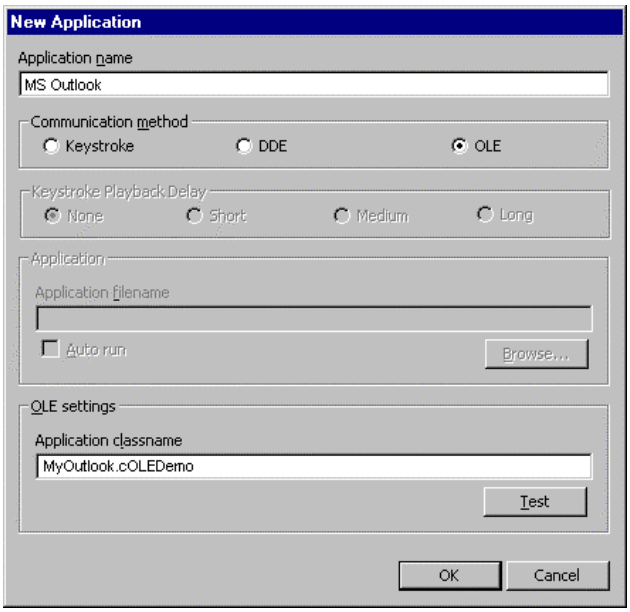


3 Type a name for this trigger.

4 Check the **Notify browser after firing** checkbox only if you are using agentCompass.

5 Click **Next** to move to the *Trigger Wizard Step 2* dialog box.

6 Click **New** to move to the *New Application* dialog box.



7 Type the Application name, Communication method, and Application classname. An example is shown below.

Example of application information

Application name	MS Outlook
Communication method	OLE
Application classname	MyOutlook.cOLEDemo

Note: The Application classname field activates when you select the OLE communication method. In this example, the OLE Automation Server is called Myoutlook, and the interface described above is exposed in the class cOLEDemo.

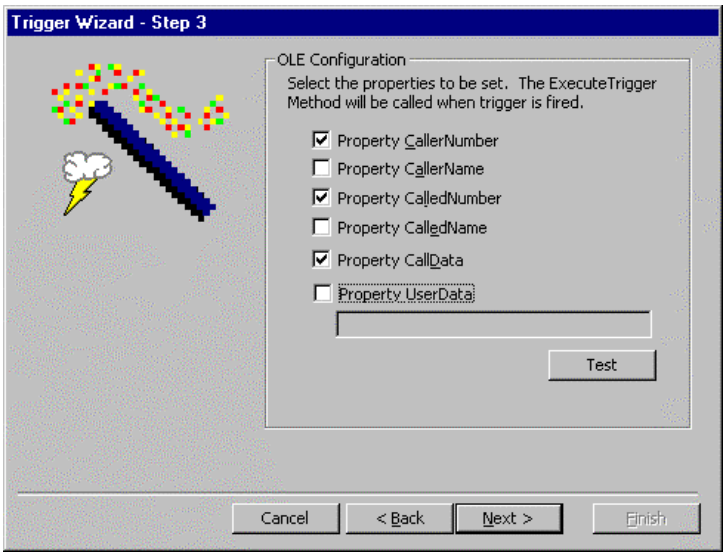
8 Click **Test** to show the *Verify Classname* dialog box. This verifies that the Application classname can be found in the Registry.

9 Click **OK** to return to the *New Application* dialog box.

10 Click **OK** to move to the *Trigger Wizard – Step 2* dialog box.

11 Highlight the application BrowserAnswer.

12 Click **Next** to move to the *Trigger Wizard – Step 3* dialog box and configure the OLE properties.



13 Click **Next** to move to the *Trigger Wizard – Finished!* dialog box.

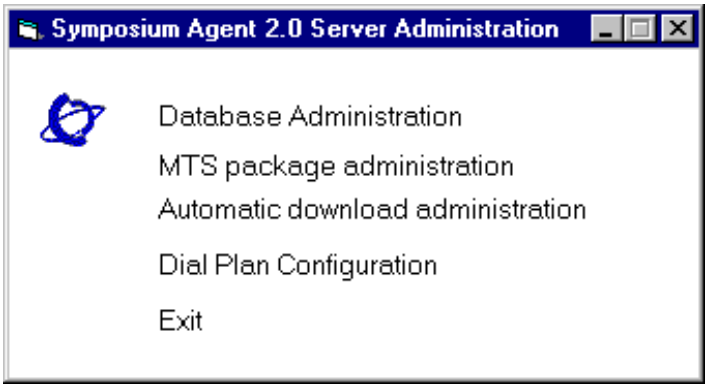
14 The dialog box summarizes the information for the trigger. Click **Finish** to open the *Call Rules Explorer* window and begin the next procedure.

Populating the SA clients with an OLE Automation Server

New or updated OLE automation servers must be installed on the client machines using the Symposium Agent Automatic download administration tool.

The OLE Automation DLL or EXE must be copied to the Nortel\Server\SASWDIST directory on the SA Server.

1 From the Symposium Agent 2.3 Server Administration menu, select Automatic download administration.

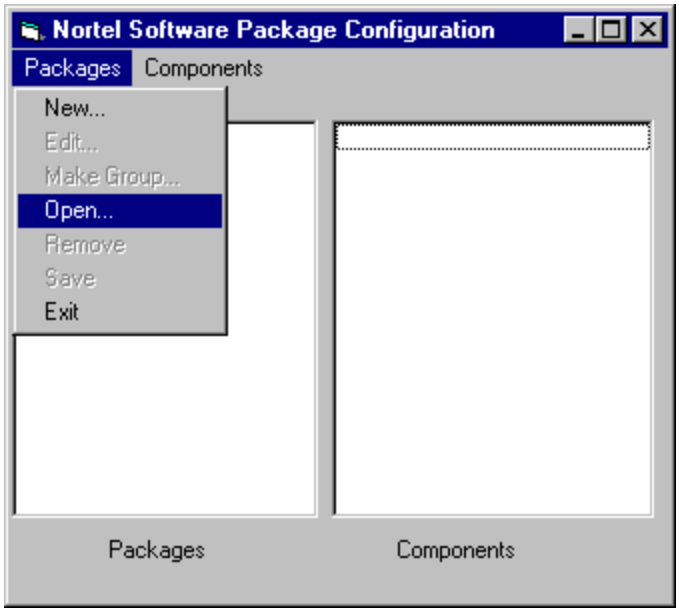


The short cut for this menu can be found on the SA Server desktop.

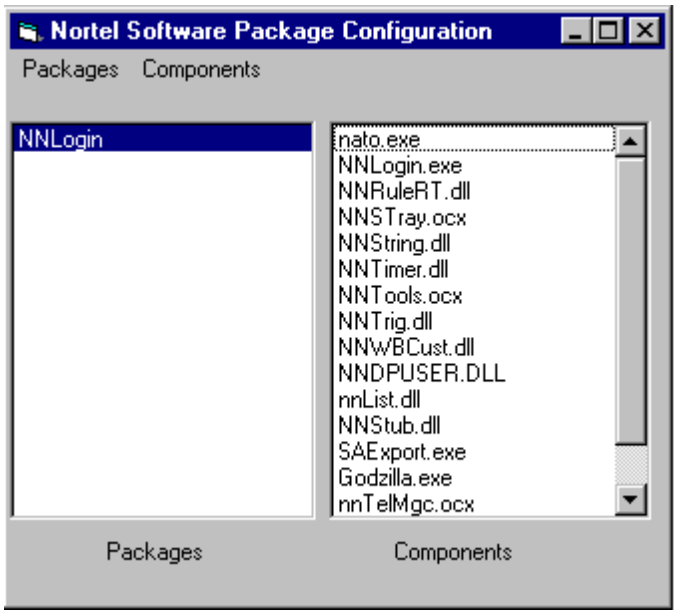
2 In the main dialog window, select Open from the Packages menu.

3 Open the file control file nnlogin.lst.

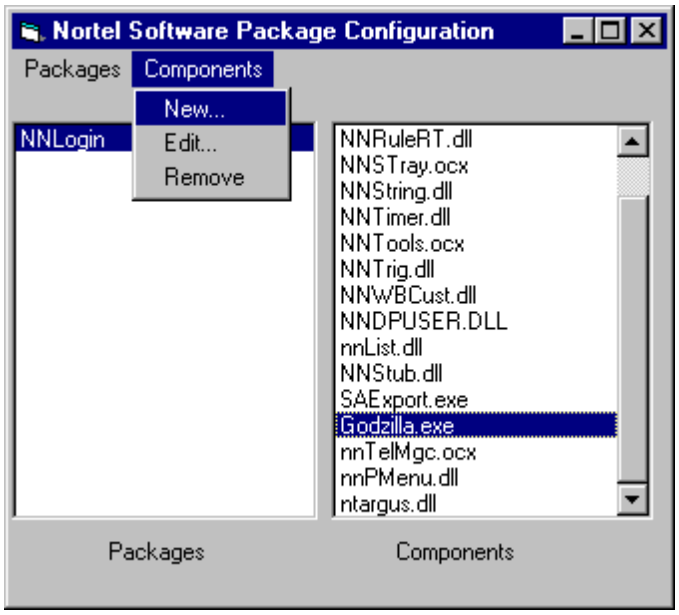
This file is located in the Nortel\Server\SASWDIST directory.



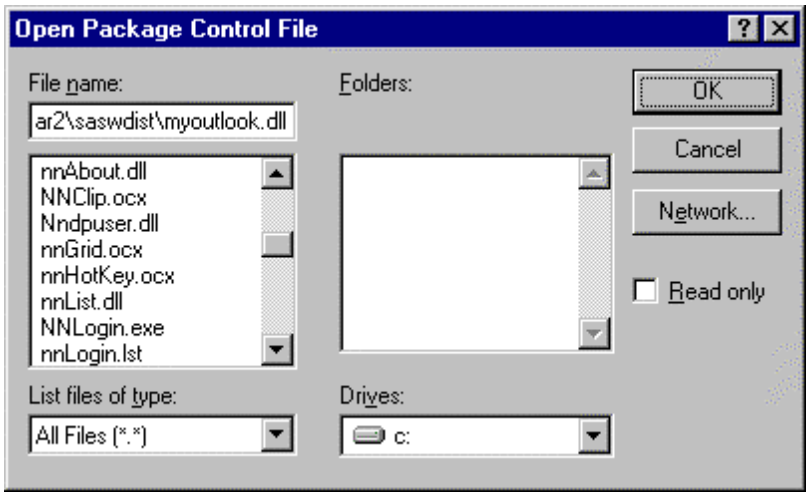
A list of components appears in the main dialog list box.



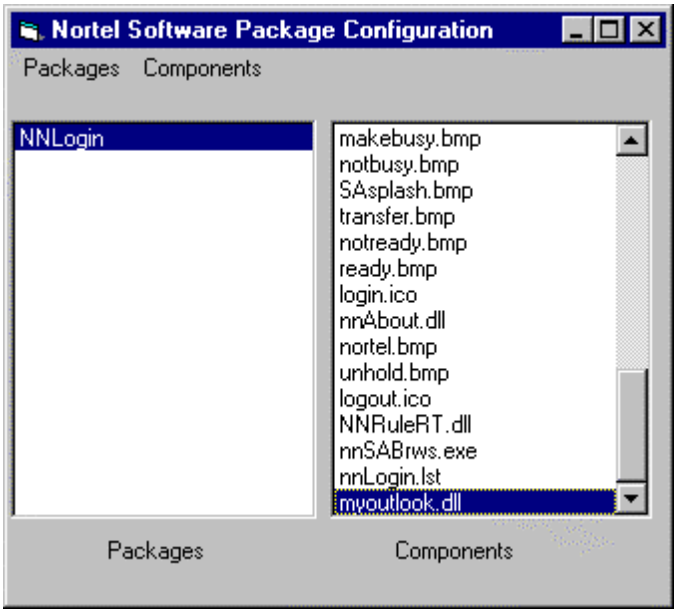
4 From the Components menu, select New.



5 Locate the new component and add it to the components list.



The file will be listed in the component listbox on the main dialog.



6 Select the newly added component. From the Components menu select Edit.

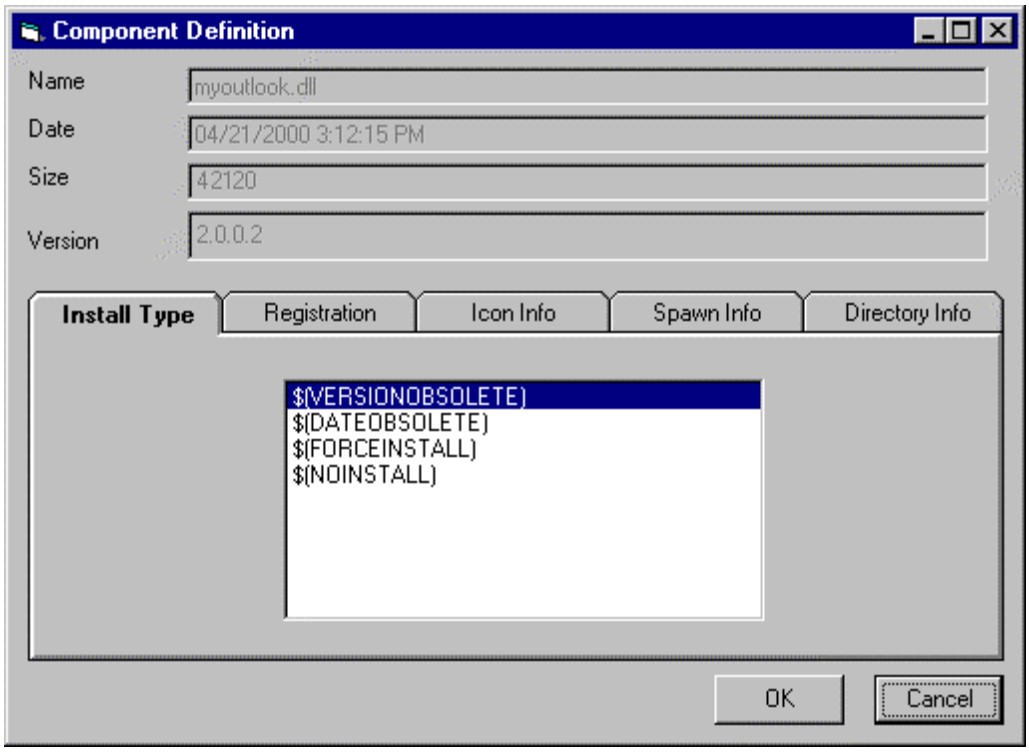
The tabbed dialog box “Component Definition” appears.

To add the new OLE automation object

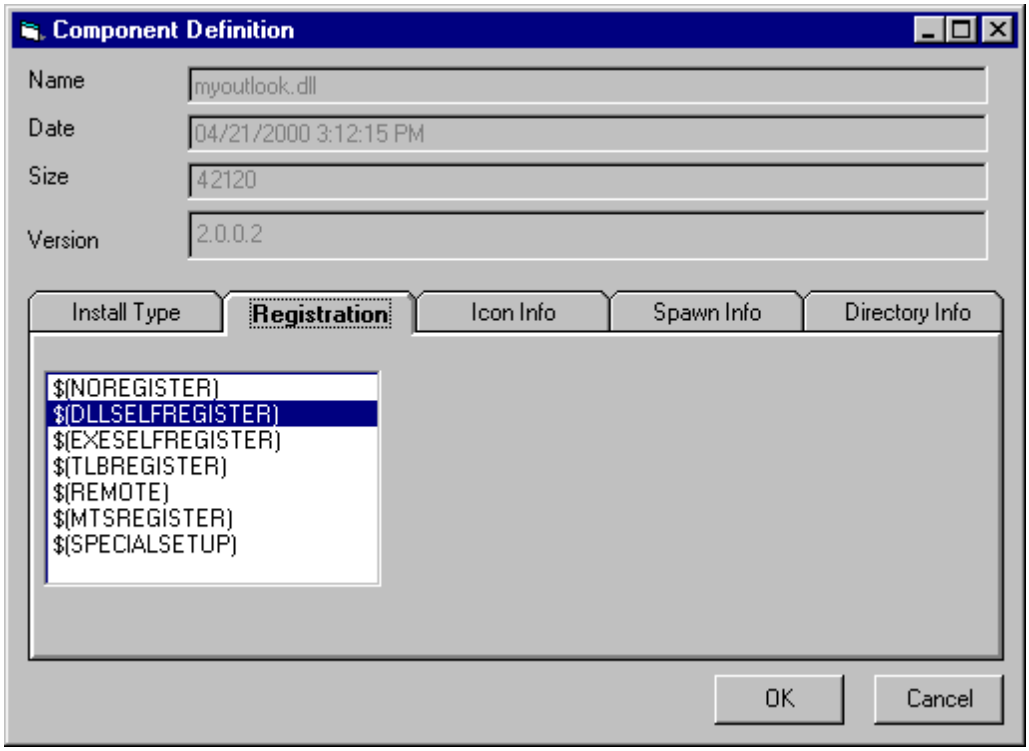
1 Enter the Install Type.

This determines how the component will be installed. If the component has a version number you can select VERSIONOBSOLETE. Otherwise, select DATEOBSOLETE.

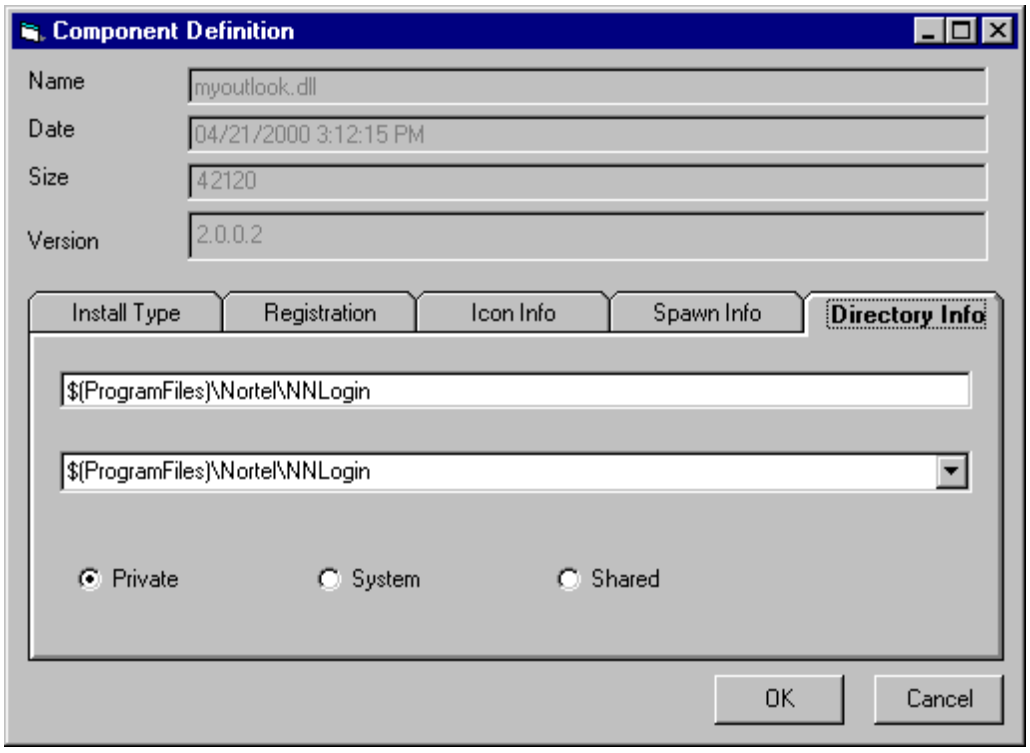
If you select FORCEINSTALL, the component will be downloaded to the server every time the launcher program is run on the client regardless of whether the component has changed.



2 Select how the component is to be registered on the Registration tab. If the component is an EXE select EXESELFREGISTER. If the component is a DLL then select DLLSELFREGISTER.



3 On the Directory Information tab indicate the location where this new component is to be installed on the client machine. The default is the ProgramFiles\Nortel\NNLogin directory.



4 Select the Private option.

5 Click OK to save settings.

6 Save the nnlogin.lst control file by selecting the Save option under the Packages menu.

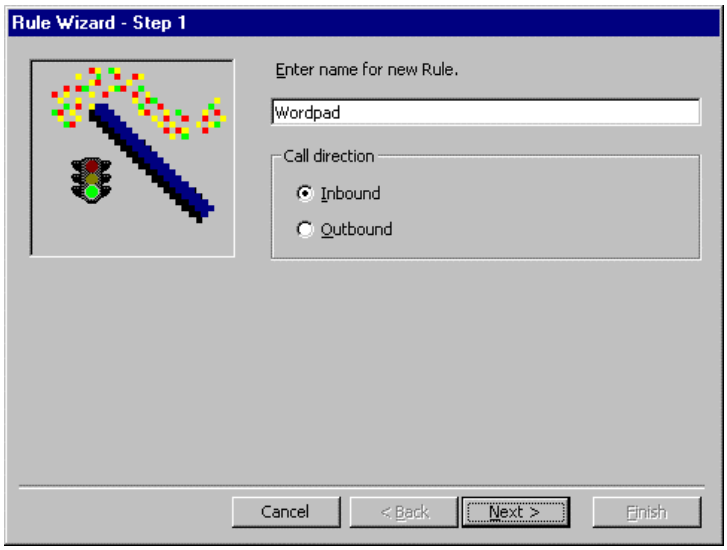
The next time the client application (nnlaunch.exe) is executed, this component will be downloaded to the client.

Setting up a call rule

To set up a call rule

1 In Call Rules Explorer, click **Rules Wizard** on the **Tools** menu. (You can also open the Rule Wizard by right-clicking the Rules icon and selecting New Rule.)

The *Rule Wizard – Step 1* dialog box opens.

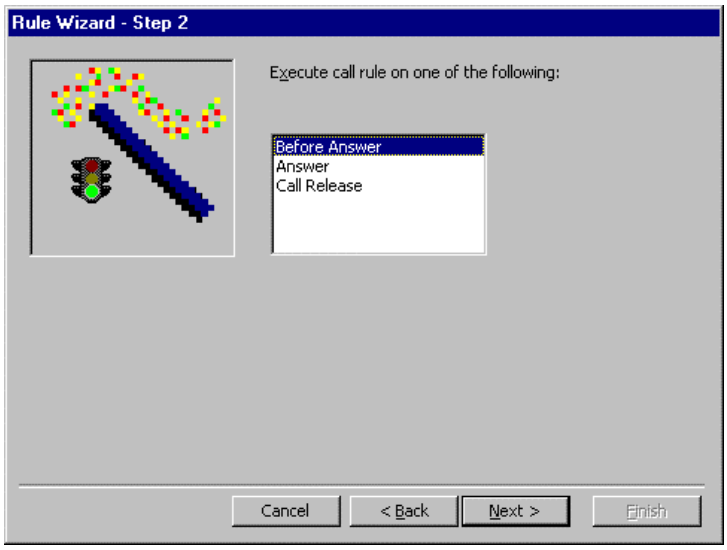


2 Type a name for this call rule. The names used in the example are shown for each browser.

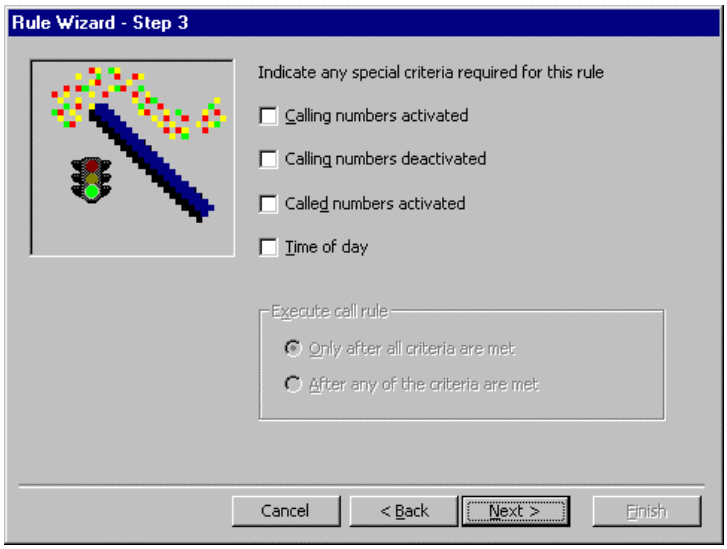
3 Click **Inbound** or **Outbound** to designate if this rule is for an inbound or outbound call.

4 Click **Next** to move to the *Rule Wizard – Step 2* dialog box and select when the call rule should execute. These options are available for inbound calls:

Before Answer	The rule runs when a new call arrives.
Answer	The rule runs when a call has been answered.
Call Release	The rule runs when the call disconnects.



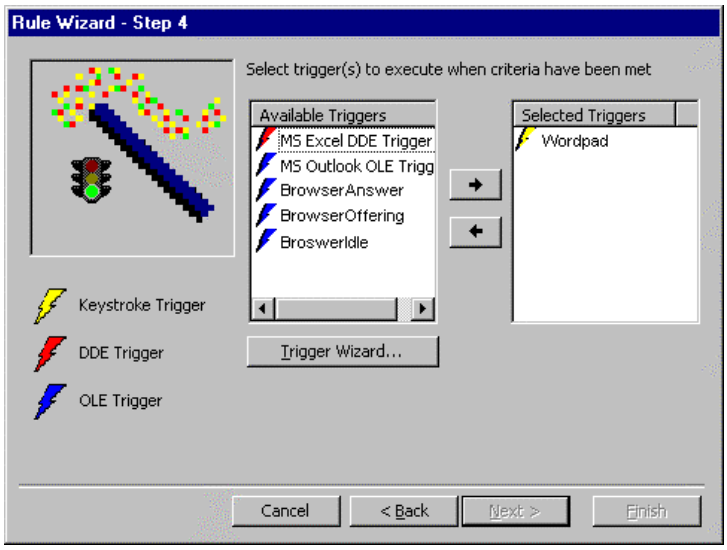
5 Click **Next** to move to the *Rule Wizard – Step 3* dialog box.



6 Rule Wizard – Step 3 presents a list of special criteria that you can select for a call rule. Criteria controls what calls will be affected by the actual trigger. If you do not select a criteria for this call rule, all calls that appear for this rule are affected.

Select criteria if required. For further information, refer to the Symposium Agent System Administrator's online Help by clicking the shortcut on the Symposium Agent Server desktop.

7 Click **Next** to move to the *Rule Wizard – Step 4* dialog box.



8 From the Available Triggers list, highlight the trigger you want to execute.

9 Click the right arrow button to move the highlighted trigger to the Selected Triggers list. When all triggers desired for this call rule have been selected, click **Next**. *The Rule Wizard – Step 5* appears.

10 The *Rule Wizard – Finished!* dialog box appears, summarizing the call rules configuration. Click **Finish**. The *Call Rules Explorer* window opens displaying the new call rule.

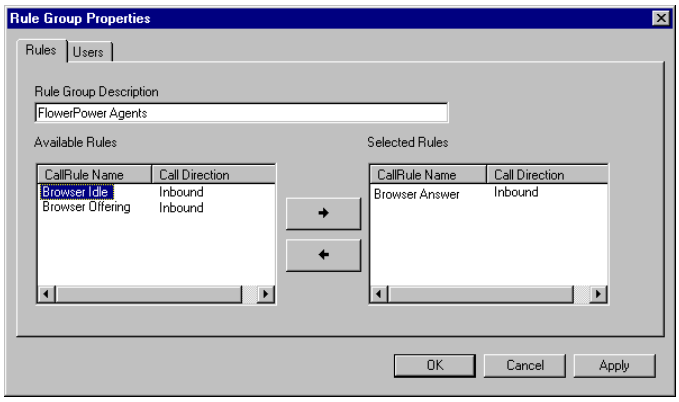
11 Select **File/Close** to exit Call Rules Explorer.

Setting up a call rules group

In Symposium Agent, agents must be associated with one or more call rules groups in order to enable any call rules.

To set up a call rules group

1 In Call Rules Explorer, click **Add Rule Group** on the **Tools** menu. (You can also open the Rule Group Properties dialog box by right-clicking the Rule Groups icon and selecting Add Rule Group.



2 On the Rules tab, in Rule Group Description, type a name for the group.

3 From Available Rules, select call rules to apply to this group and move them to the Selected Rules list.

4 Select the **User** tab. Select users to add to this group.

5 Select **OK** to exit.

Chapter 3

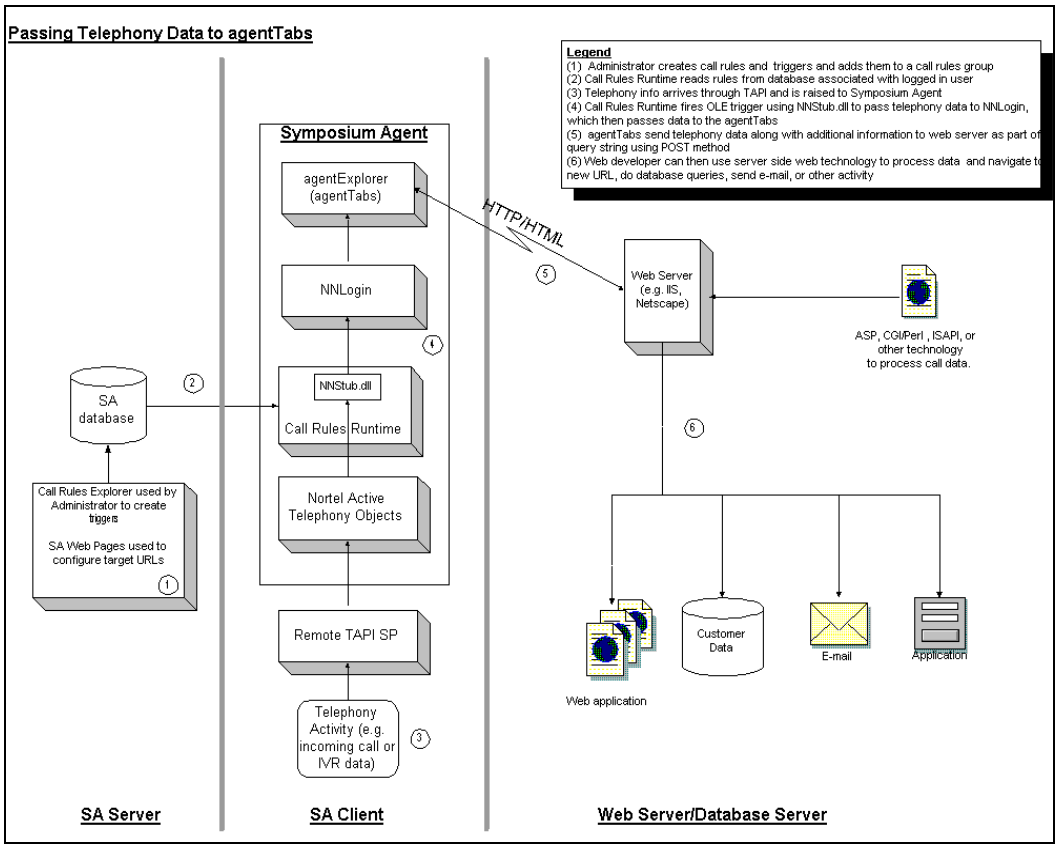
Using Agent Compass

In this chapter

Symposium Agent data flow in agentCompass	34
Query strings	35
Web server setup	37
Client setup	38
Setting up an agentExplorer call trigger	39
Setting up and enabling agentCompass call rules	41
Setting up telephony-enabled tabs for agents	42
Testing the call rule	44

Symposium Agent data flow in agentCompass

The following diagram shows how data is delivered to the agentTabs.


Passing Telephony Data to agentTabs
Legend
(1) Administrator creates call rules and triggers and adds them to a call rules group
(2) Call Rules Runtime reads rules from database associated with logged in user
(3) Telephony info arrives through TAPI and is raised to Symposium Agent
(4) Call Rules Runtime fires OLE trigger using NNStub.dll to pass telephony data to NNLogin, which then passes data to the agentTabs
(5) agentTabs send telephony data along with additional information to web server as part of query string using POST method
(6) Web developer can then use server side web technology to process data and navigate to new URL, do database queries, send e-mail, or other activity
Symposium Agent
agentExplorer (agentTabs)
NNLogin
NNStub.dll
Call Rules Runtime
Nortel Active Telephony Objects
Remote TAPI SP
Telephony Activity (e.g. incoming call or IVR data)
SA database
Call Rules Explorer used by Administrator to create triggers
SA Web Pages used to configure target URLs
HTTP/HTML
Web Server (e.g. IIS, Netscape)
ASP, CGI/Perl , ISAPI, or other technology to process call data.
Web application
Customer Data
E-mail
Application
SA Server
SA Client
Web Server/Database Server

During initialization, Call Rules Runtime retrieves the call rules and triggers from the Symposium Agent database for the currently logged in user. The call is passed to the TAPI interface, which raises the telephony data to the Call Rules Runtime. Rules Runtime initiates the correct trigger based on call criteria such as call state (on Answer). The OLE trigger raises the call data to the SA client browser (agentExplorer) using a component called nnstub.dll. This DLL communicates with NNLogin, which passes the telephony information to the agentExplorer. The agentExplorer formats the telephony information and additional information as a query string. The query string is appended to the web page URL associated to a particular tab and sent to the web server using the POST method.

Query strings

The example below shows a sample URL with attached query string:

```
http://MyWebServer/NortelWeb/CallAnswer.asp?CallerNumber=1411&CalledNumber
=2310&CallData=1234&CallID=1&State=Answer&Extension=2310&LineName=4.0.4.
14&LoginID=mjones&LanguageID=1033&TrunkInfo=112312&Timezone=5&Version=
2.0&ACDAgentID=5342&CallerName=&CalledName=
```

Note: In this example, CallerName and CalledName are blank. When a particular piece of information is not provided to the agentExplorer, the parameter names are still attached to the query string.

The following table provides a brief description of the URL plus query string shown above.

Query string explanation

http://MyWebServer./NortelWeb/CallAnswer.asp	Each agent tab that has been telephony-enabled has a URL assigned to it for one or more of the following call states: on release, on answer, and on offering.
?	A question mark indicates the beginning of attached data (the query string follows).
CallerNumber=1400	This is an attribute/value pair. Each piece of telephony data passed to the agentTab is represented in this manner. As noted above, telephony attributes that do not have a value are still passed in the query string and take on the form “CallerName=”
&	The ampersand is the separator between attribute/value pairs.

The POST method is used to pass the telephony information to the web server. This method allows the querystring to be hidden from view. This means that the query string will not show up in the address bar of the browser.

The querystring, including URL attributes and values, has no defined length when posting data. The query string attributes are detailed in the table below. The information in the table includes additional data, denoted by an asterisk (*), provided by the agentCompass feature.

Explanation of telephony data passed to agentTabs

CallerNumber	Number of party who placed the call.
CalledNumber	Number of party or DNIS receiving the call.
CallData	Attached dat, such as IVR data.
CallID	The callID is the unique identifier for a call in a networked environment.
State	This is the call state (Connected, Idle, Offering).
Extension	The extension on which the call arrived.
LineName	The line name on which the call arrived.
LoginID *	This is the Symposium Agent login ID.
LanguageID *	This is the local ID stored in the database for this user.
TrunkInfo	TrunkInfo consists of two numbers: the trunk group and number for calls outside the local switch. This information might be used by voice recording equipment that sits on a trunk and records calls based on trunk group and member numbers.
Timezone **	Bias from GMT for the various timezones. E.g. +5 or –5.
Version *	This is the Symposium Agent build number
ACDAgentID *	This is the ACD agent login ID (e.g. 5432)
CallerName	This is the caller name. Since this does not always show up, it was placed at the end of the query string.
CalledName	This is the called name. Since this does not always show up, it was placed at the end of the query string.

* These items are not telephony data but originate in the Symposium Agent database.

** This item is derived from the client computer.

Web server setup

You set up a web server and create web pages using technology that will process the call data. After the web pages have been tested, they are published on the web server. The web server can use call data to trigger a jump to a new web page, perform database queries, or some other activity.

You must complete the following tasks before starting the client setup:

- Set up a web server according to the product's documentation. The example in this guide uses Microsoft's Internet Information Server (IIS). (The Symposium Agent 2.3 server requires IIS. You can choose to use the Symposium Agent server as your web server, but be aware that large numbers of hits on IIS may degrade Symposium Agent server performance.)

- Create an applicable alias web directory; the example uses
  c:\Nortel\Server\SAWebApp

- Verify that the web directory has Read and Scripts permissions

Symposium Agent provides a simple ASP page that demonstrates parsing and display of the query string data. This page is located on the Symposium Agent Server in
  c:\Nortel\Server\SaWebApp\Samples\nnweb.asp.

Client setup

The administrator creates agentTabs for the web application in Symposium Agent Administration. These tabs define the target URL(s) and whether these URLs will receive data. Additionally, the administrator sets up call rules and OLE triggers for the agentExplorer application. These call rules are then associated with Symposium Agent users in a call rules group. The client (agent) launches the agentExplorer and Call Rules Runtime. Rules Runtime's system tray traffic light icon turns green when it initializes. The client (each agentTab) is then ready to receive telephony events.

Note: Before you can use triggers and call rules and create agent tabs, you must have set up a web server and a web directory.

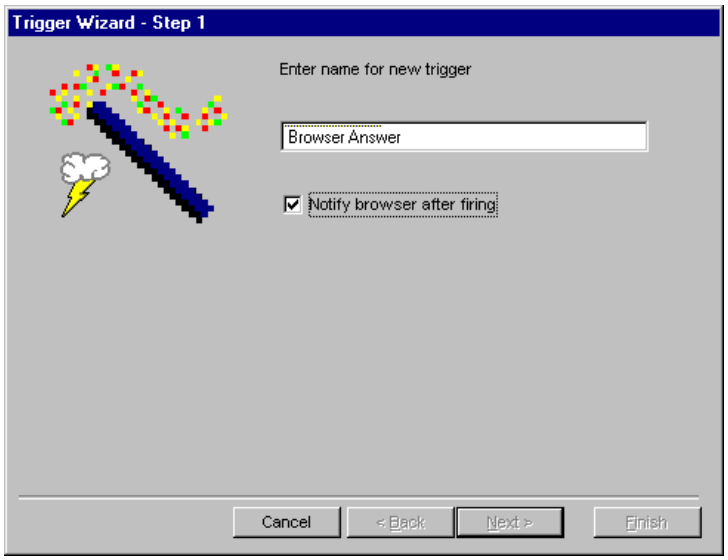
Setting up an agentExplorer call trigger

Symposium Agent installs default call rules and OLE triggers that can be used with the agentExplorer application without modification. However, these call rules or triggers may need modification to add specific call criteria such as time of day or caller number if desired. In addition, Symposium Agent provides an OLE DLL used by the OLE triggers to communicate with the agentExplorer. Follow the steps provided in *Setting up an OLE trigger*, on page 24. Items specific to agentCompass are mentioned below. If you choose to use the default triggers and call rules, skip to *Setting up a call rules group*, on page 32.

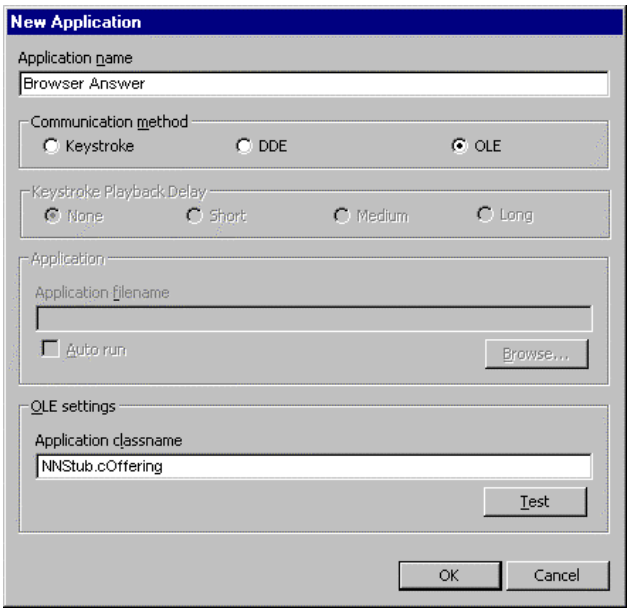
To set up a trigger for agentCompass

1 From the Trigger Wizard – Step 1 dialog box, ensure that the **Notify browser after firing** box is checked.

The *Trigger Wizard – Step 1* dialog box appears.



2 Enter a name for this trigger, then click **Next** to move to the New Application dialog box.



3 In the New Application dialog box, refer to the default OLE DLL provided by Symposium Agent. Depending upon the desired call state, use the information provided in the table below.

Example of application information

Application name	BrowserAnswer
Communication method	OLE
Application classname for call state Answer	NNStub.cAnswer *(see note below)*
Application classname for call state Offering	NNStub.cOffering
Application classname for call state Idle	NNStub.cIdle

Note: The Application classname field activates when you select the OLE communication method. In agentCompass, the Application classname is always one of NNStub.cAnswer, NNStub.cOffering, or NNStub.cIdle.

4 The OLE properties dialog box can be ignored. All properties listed on this page (if available) are sent to the browser. Set the URL property on the Tab configuration web page as described later in this chapter. Do not test from this dialog box. AgentCompass includes a sample web page (nnweb.asp) that you can use for testing through the Symposium Agent client.

Setting up and enabling agentCompass call rules

To set up a call rule

Follow the steps described in the section *Setting up a Call Rule*, on page 29.

To set up a call rule group

You can create a new call rule group, if necessary, and add call rules and agents to the desired call rule group.

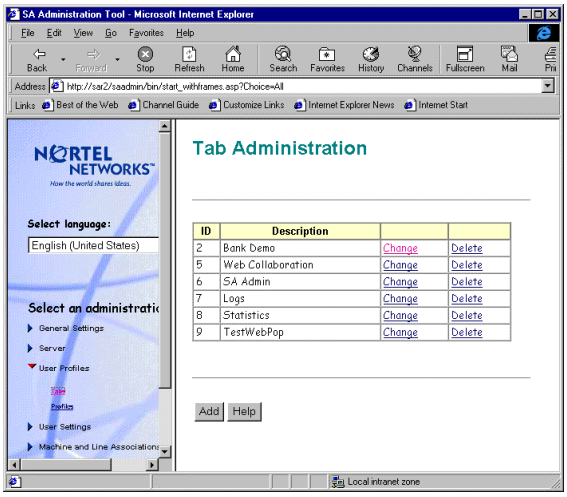
Follow the steps found in the section *Setting up a call rules group* on page 32, and add the agentCompass call rules to a call rule group.

Setting up telephony-enabled tabs for agents

The following is a quick summary of how to create an agentTab that is telephony enabled. For detailed information, refer to the online Help.

To create a tab

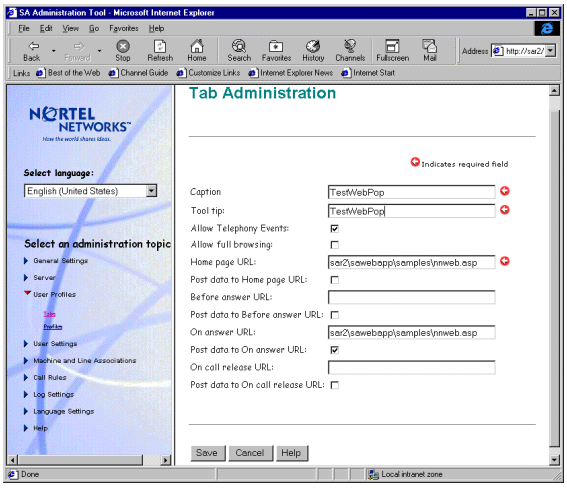
1 Go to the Symposium Agent Administration web pages.



2 In the index to the left, select **Tabs** under User Profiles administration.

3 In the right frame, press the **Add** button on the Tabs Administration page.

4 Enter the tab caption and tooltip.



5 Select Allow Telephony Events.

6 Enter the URL of the home page for this tab.

7 Enter the URL for one or more of the call states.

8 Select the Post Data to On Answer URL.

9 Click **Save** to save this tab's settings.

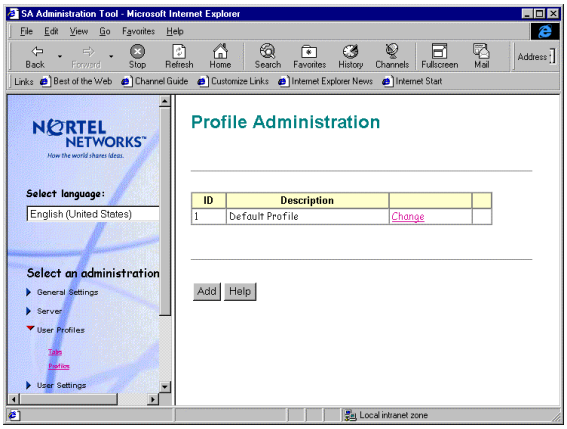
To add this tab to a user profile

For more detailed information on this procedure, refer to the online Help.

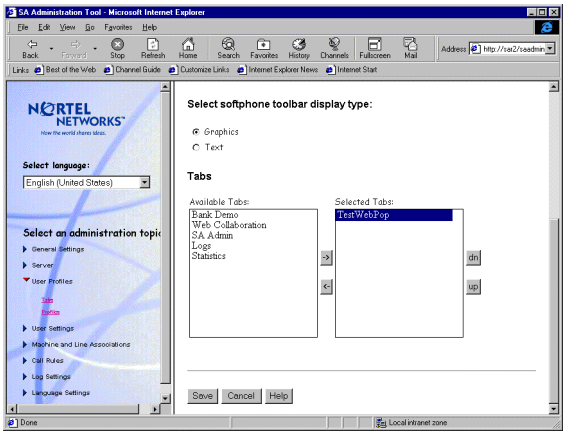
1 Go to the Symposium Agent Administration web pages.

2 In the index to the left, select **Profiles** under User Profiles administration.

3 Select **Change** to change an existing profile.



4 Or, click **Add** to create a new profile. (Refer to the online Help.)



5 Select the tabs to be added to this profile.

6 Click **Save** to save the profile.

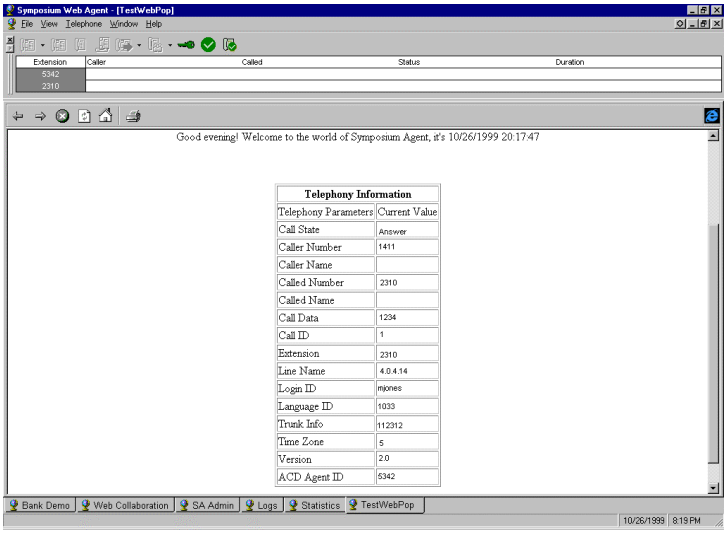
Note: Some web servers may not know how to handle posted data attached to an HTML page. In some cases, you may want to trigger navigation to an HTML page but do not want to pass any data along. In these situations, make sure you uncheck the Post Data flag for these pages.

Testing the call rule

When you have completed the steps for setting up client tabs, use the test web page nnweb.asp to ensure that data is being passed to the agentTabs and then to the web server.

To test the call rule

1. Launch the client application.
2. Ensure the tab you defined is visible as an agentTab.
3. Make a call to the client.
4. Answer the call on the client.
5. The defined tab should update with call information.



Chapter 4

Getting Started with Symposium Agent Reporting

In this chapter

Overview	46
Configuring the database access	48
Defining and ODBC system DSN	52
Database Entity Relationship Diagram	55
Data Dictionary	57
Sample Report using Crystal Report	70

Overview

The Symposium Agent database is an open database that is accessible from applications that support Open Database Connectivity (ODBC) and Structured Query Language (SQL).

The Symposium Agent administrator can configure the Symposium Agent client with three web page views of the reporting data found in the Symposium Agent database. These web pages, referred to as agentTabs, are for call logs, ACD activity logs, and call statistics. You can create your own reports for log information and call statistics using Crystal Reports or other applications that are ODBC or SQL compliant.

You can use the data in the database in several ways including:

- Importing into a spread sheet for manipulation
- Generating customized reports using report generation tools such as Crystal Reports
- Creating web based reports using server side scripting and database access objects

Skills you need

If you want to create reports of the call log and statistics data in a format other than those that appear on the web pages, you will need to have familiarity with the following:

- Crystal Reports or other reporting tool
- Understanding of server-side scripting if creating a web based report
- Structured Query Language (SQL)

Storage duration

The duration for which statistics are stored in the Symposium Agent database is configurable through the Symposium Agent Administration web pages. See the help panels available through the Symposium Agent Administrator web pages details.

Type of data collected

Statistics are collected only if the Symposium Agent Call Monitor service is running on the SA server. The types of statistics collected include call information and ACD agent activity. The call monitor service can be started from the services

control panel applet or the SA Administration pages. See the SA installation guide or the help panels available through the Symposium Agent Administrator web pages.

Before you start

You need the following to get started:

- Sybase open client must be installed
- ODBC must be installed
- ODBC System Data Source Name (DSN) for Sybase must be defined
- The required privileges to access the Symposium Agent Database

Sybase

If you are working from the Symposium Agent Server then the necessary Sybase components, including ODBC drivers for Sybase, have already been installed. In addition, the ODBC System DSN is already defined.

If you require access to the Symposium Agent database from a remote client (on a PC other than the SA Server) then the ODBC drivers for Sybase need to be installed.

To install ODBC drivers

1 Install the Sybase Open Client components

This product is available through the Sybase ASE server setup program. Select the Open Client Runtime Option.

2 Using the DSEdit utility configure an entry for the Sybase Server located on the Symposium Agent Server.

Notes:

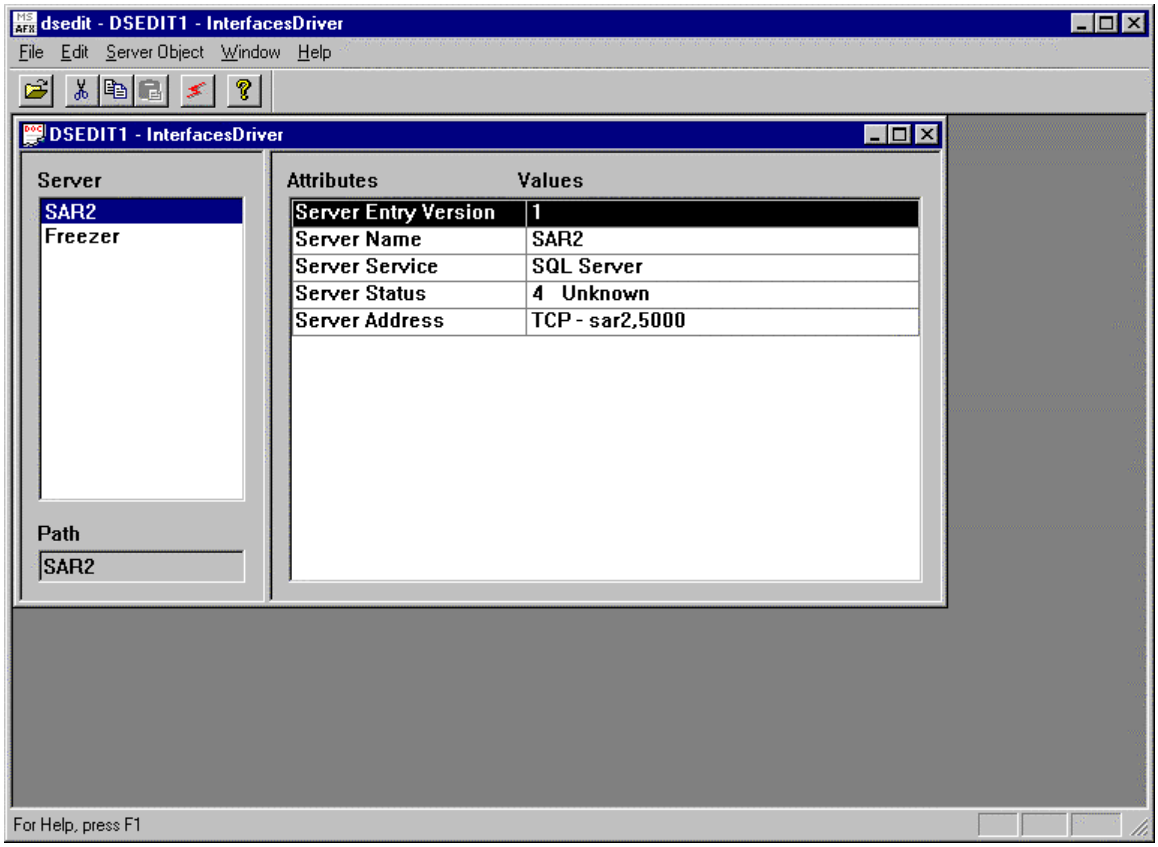
- All references to Sybase are for Sybase Adaptive Server 11.5.1.
- Nortel Networks does not provide licenses for the Sybase Open Client outside of those components installed on the Symposium Agent Server. It is up to the customer to obtain valid licenses for the Open client components if accessing the Symposium Agent database from another PC.

Configuring the database access

This section provides instructions on configuring a Sybase server in Sybase Open client and verifying your PC's connection to the Symposium Agent Database.

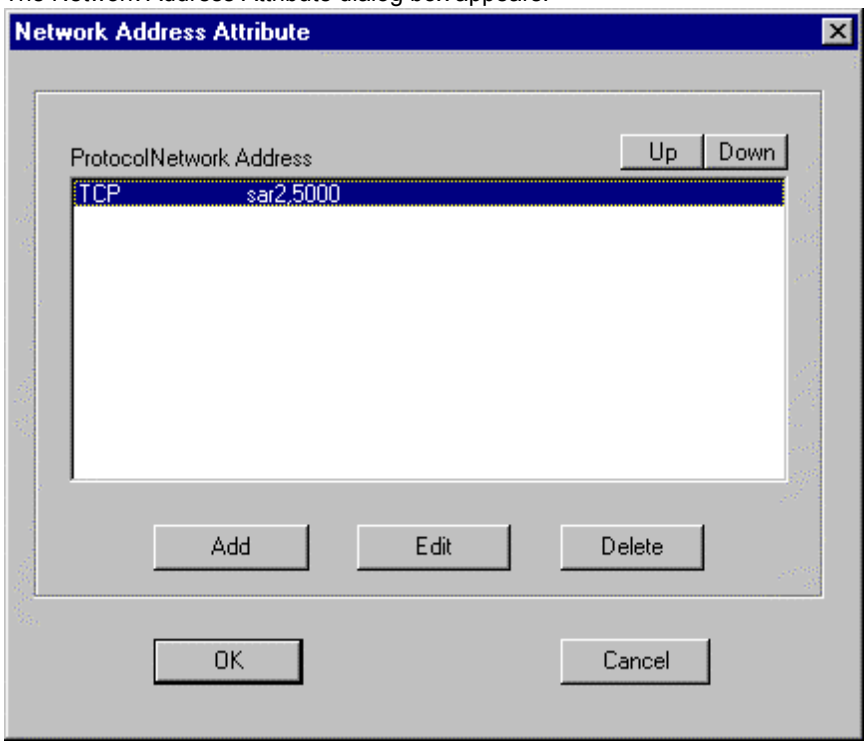
Defining a connection the Server

1. From the Windows Start menu, choose Programs->Sybase->DSEdit.
2. Select "InterfacesDriver" from the listbox. You will then see a list of currently defined servers.

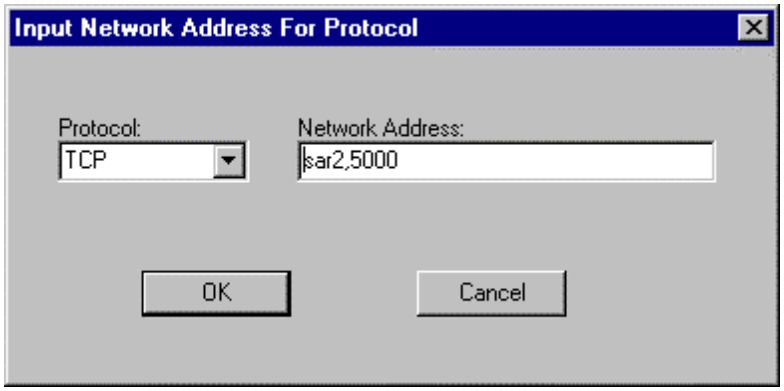


3. If your server is not defined then select "Add" from the "Server Object" menu. Then perform the following steps:
4. Enter a server name. It will be added to the list of servers.
5. Double click on the Server Address attribute of the newly added server name.

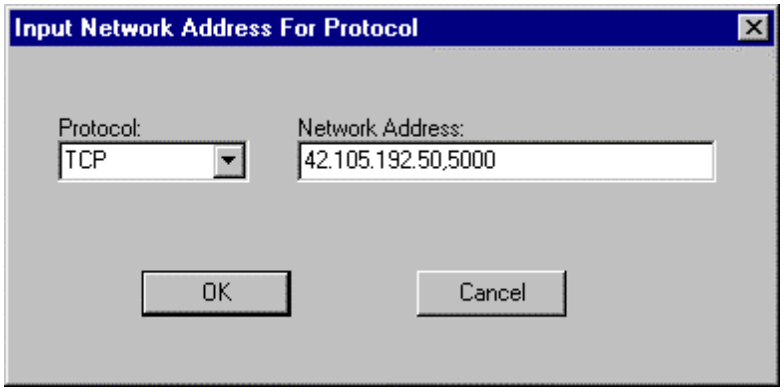
The Network Address Attribute dialog box appears.



3 Click “Add” to set the Network Address and Protocol.



The name “sar2,5000” in the dialog box above represents the network address followed by the port used by the Sybase service. You can also specify the IP addresses in place of the server name as in the example below.



Verifying the server connection

To verify the server address

1. Verify that the server has the correct network address and protocol defined.
2. If the server address is not correct, double click on the attribute to bring up the “Network Address Attribute” dialog box.
3. Click Edit to view the address details and make the necessary changes.

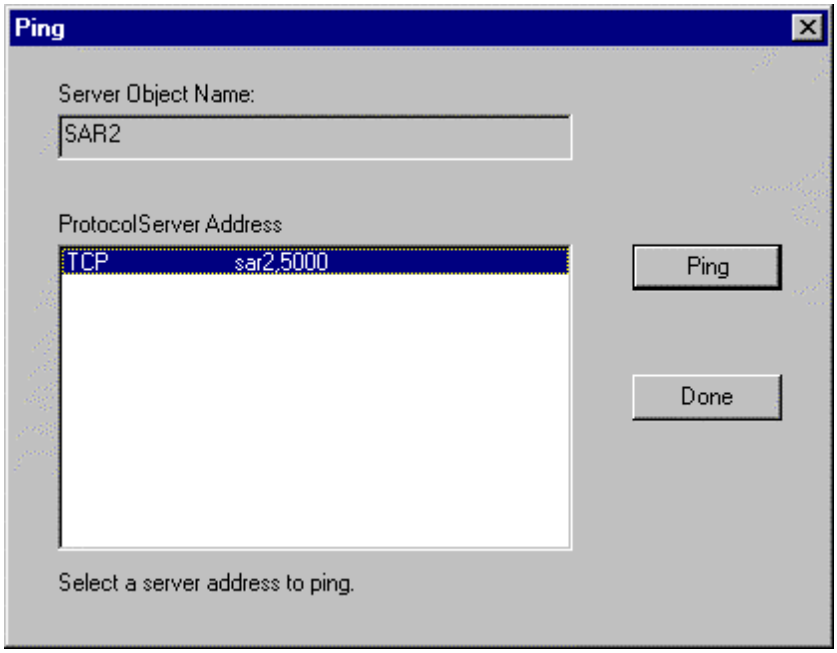
To verify the server connection

You can verify the server connection by using the Ping utility that comes with DSEdit.

1. From the DSEdit main dialog box, select the name of the server, and right click on the server name.
2. Select “Ping Server” from the popup menu.

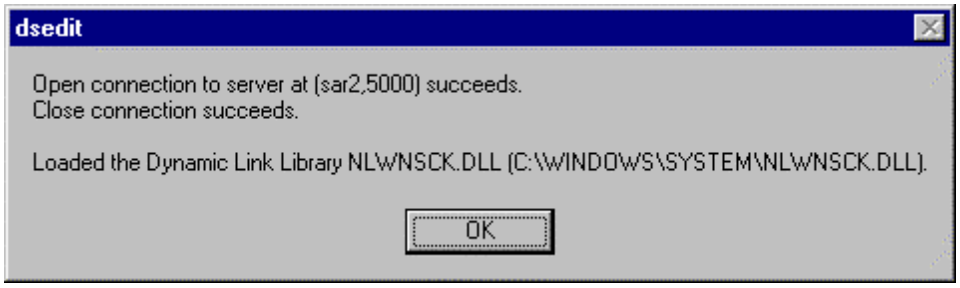
   You can also access the “Ping Server” utility from “Server Object” menu.

The Ping dialog box appears.



4 Click Ping to ping the Sybase Server.

If the connection is valid, the results look something like this:



If the ping fails then check the server network address. You may need to specify an IP address if you are currently using the server name.

Defining an ODBC System DSN

Because the Symposium Agent database is ODBC-compliant, the PC from which you access it must have ODBC installed. It must also have a System DSN defined for Sybase.

Note: Nortel Networks does not install ODBC software or setup a System DSN on client machines.

To install ODBC

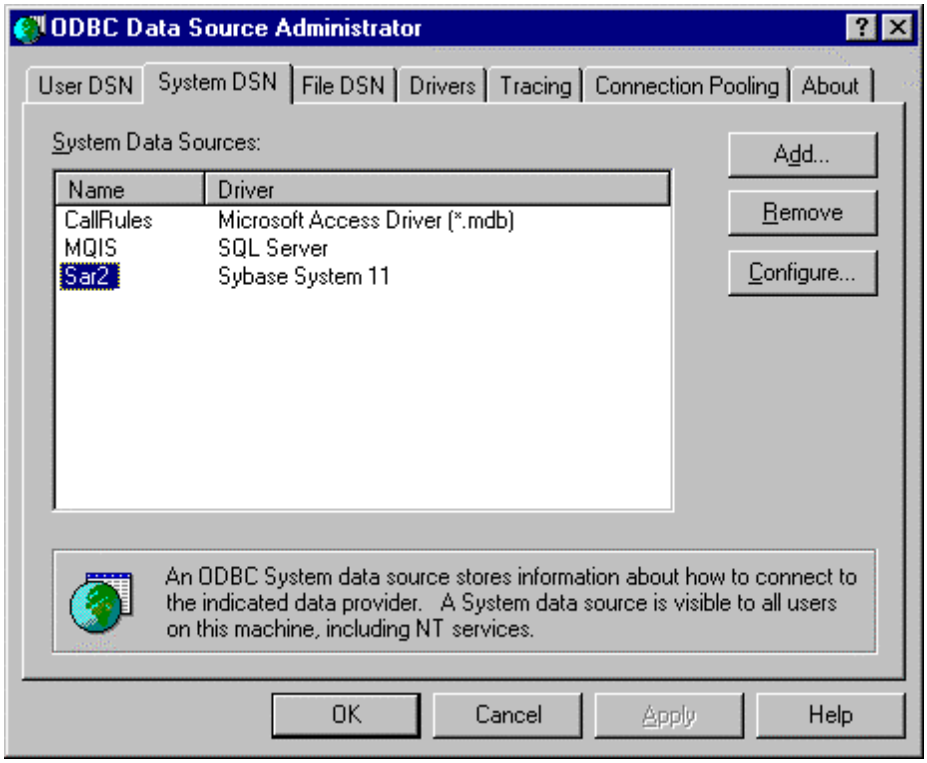
ODBC should already be installed on your PC. However, if ODBC is not installed, then do the following.

1 Install ODBC.

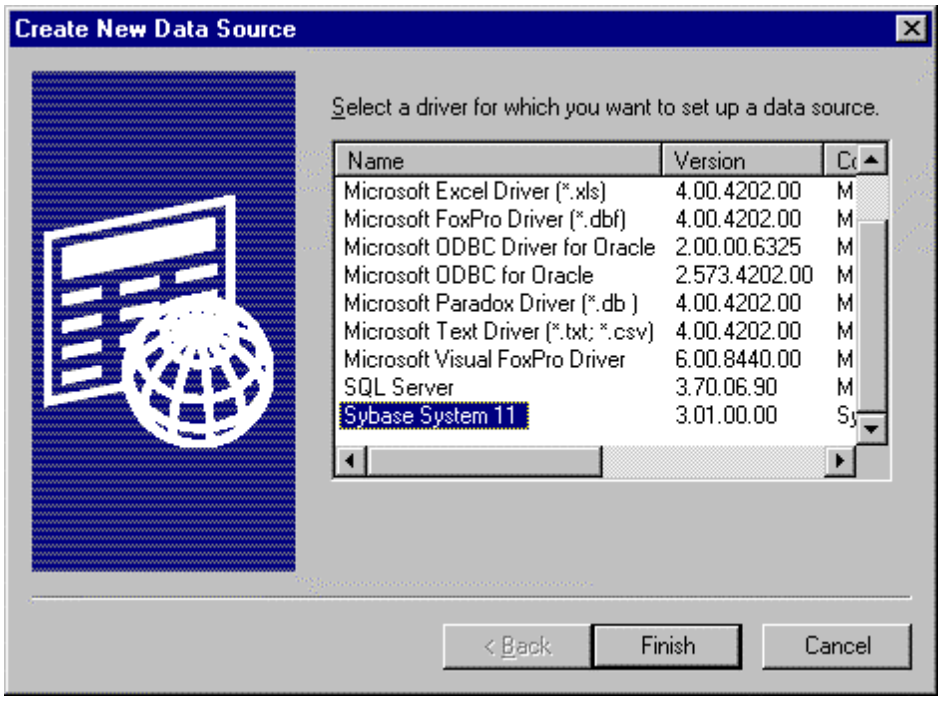
ODBC is part of the Microsoft Database Access Components (MDAC). It is usually installed with Microsoft Windows. See the SA installation guide for the MDAC version to install. You can find a copy of MDAC (mdac_typi.exe) in the Nortel\Server\SASWDist folder on the SA server.

To create a System DSN for Sybase

1 From the Control Panel click on ODBC Data Sources. Select the System DSN tab.



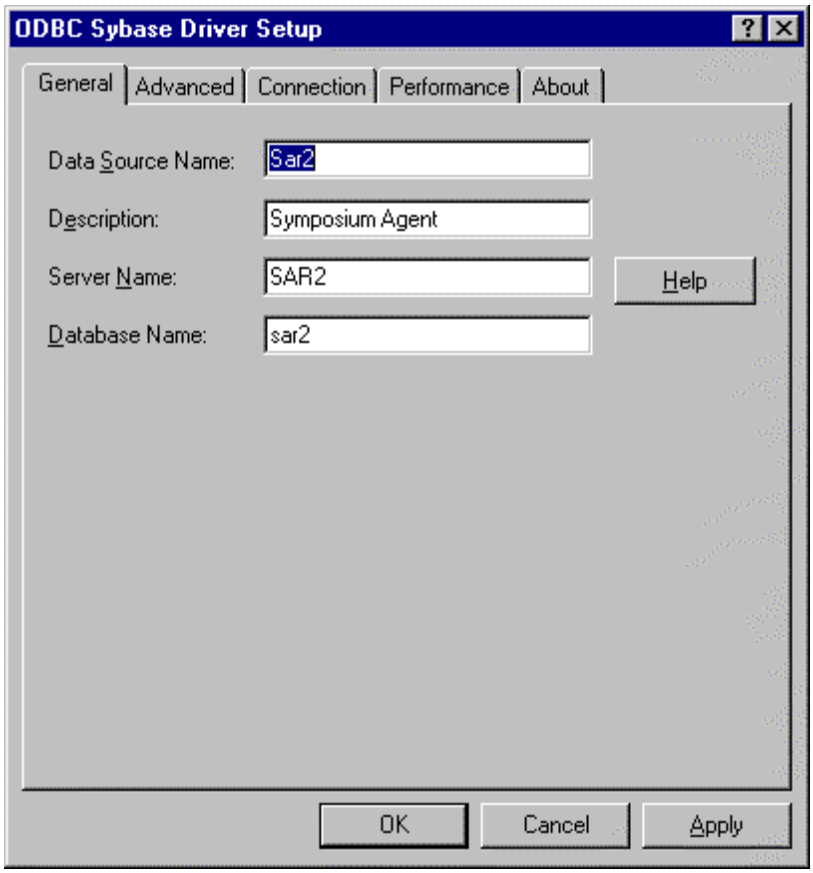
2 Click on “Add” to create a new System DSN.


Create New Data Source
Select a driver for which you want to set up a data source.
Name
Version
Microsoft Excel Driver (*.xls) 4.00.4202.00
Microsoft FoxPro Driver (*.dbf) 4.00.4202.00
Microsoft ODBC Driver for Oracle 2.00.00.6325
Microsoft ODBC for Oracle 2.573.4202.00
Microsoft Paradox Driver (*.db) 4.00.4202.00
Microsoft Text Driver (*.txt; *.csv) 4.00.4202.00
Microsoft Visual FoxPro Driver 6.00.8440.00
SQL Server 3.70.06.90
Sybase System 11 3.01.00.00
< Back
Finish
Cancel

3 Select the Sybase ODBC Driver.

4 Enter the following information on the “ODBC Sybase Driver Setup” dialog.

Data Source Name:	Sar2
Description:	Symposium Agent
Server Name:	<Symposium Agent Server Name e.g. SAR2>
Database Name:	sar2



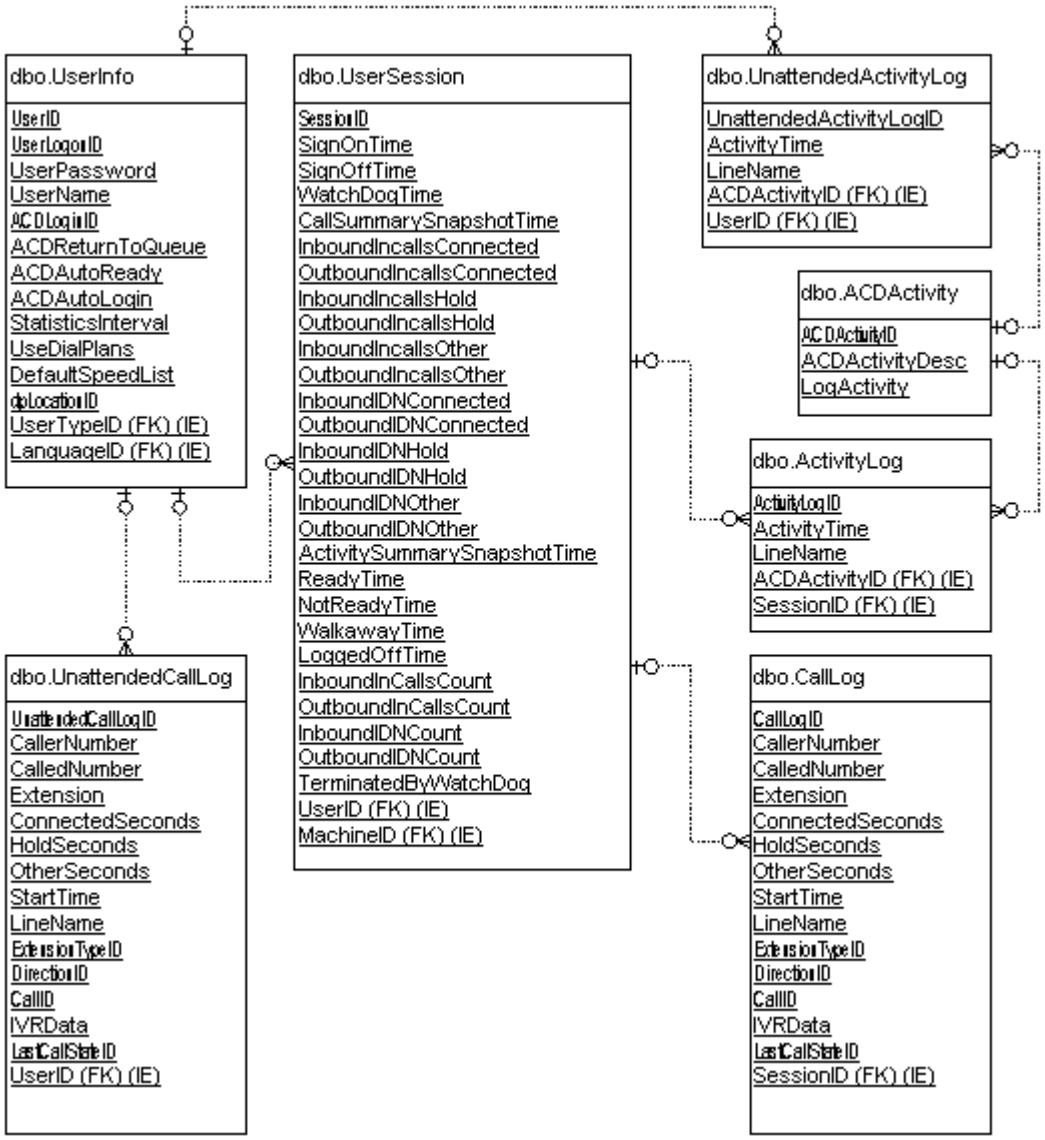
Access Rights

You will need the necessary access rights to get to the Symposium Agent database. Database connectivity requires you have access to the appropriate SA database user account. The following username and password can be used to access a read only view the tables necessary for reporting.

User name: saview
Password: saview

Database Entity Relationship Diagram

The following Entity Relationship Diagram (ERD) shows the relationship of the tables housing statistical data in the Symposium Agent database.



Information of interest for call and agent activity reporting are in the following tables:

- CallLog
- UnattendedCallLog
- ACDActivity

- ACDActivityLog
- UnattendedACDActivityLog
- UserSession
- UserInfo

Data Dictionary

Overview

This section describes the data that is available from the Symposium Agent database.

Data types

The database types used by the Symposium Agent database are as follows. The following table is an excerpt from "Sybase SQL Server 11 Unleashed" by SAMS Publishing.

Datatype	Range of Values	Size in Bytes
Character Datatypes		
char[(n)]	1<=n<=255	N
varchar[(n)]	1<=n<=255	Data length, up to n
Text	BLOB up to 2,147,483,647 chars	16 + multiple of 2K
Date Datatypes		
Datetime	Jan 1, 1753 to Dec 31, 9999. Accuracy to 3 millisecond interval.	8
Smalldatetime	Jan 1, 1990 to June 6, 2079. Accuracy to minute.	4
Logical Datatypes		
bit	0 or 1	1
Numeric Datatypes		
int	± 2,147,483,647	4
smallint	± 32,767	2
tinyint	0 through 255	1
float[(precision)]	± 2.23E-308 (very precise) ± 1.79E308 (very big)	8
real	± 1.18E-38 (very precise) ± 3.40E38 (very big)	4
numeric(p,s) decimal(p,s) (dec)	± 10 to 38^{th} power, p is precision (total digits, 1-38), s is scale (decimal digits <= p)	2 through 17

Table and Field descriptions

Each table and its corresponding fields (as shown in the ERD) are detailed in the following sections.

ACD Activity

Introduction

The ACD Activity table contains descriptions for the various ACD Activities. Use this table to convert an ACD activity ID into a user-friendly string.

Table

ACDActivity

Field descriptions

ACDActivityID

Description: Unique table id
Type: numeric
Length: 10

ACDActivityDesc

Description: Description of ACD Activity
Type: varchar
Length: 255

LogActivity

Description: Flag for enabling/disabling logging for an ACD activity
Type: smallint
Length: 2

Unattended ACD Activity Log

Introduction

The unattended ACD activity log contains data on unattended ACD activity. This table holds records for every agent state change for Ready, Not Ready, Login, and Logout on a unattended machine. Unattended activity is monitored on machines that are not logged into to the Symposium Agent Server.

Table

UnattendedActivityLog

Field descriptions

UnattendedActivityLogID

Description: Unique table id
Type: numeric
Length: 10

ActivityTime

Description: Date and Time ACD activity took place
Type: datetime
Length: 8

UserID

Description: unique id for user associated with ACD Activity
Type: numeric
Length: 10

ACDActivityID

Description: Unique Activity id to determine ACD Activity
Type: numeric
Length: 10

LineName

Description: Line device name on which the activity took place
Type: varchar
Length: 50

Unattended Call Log

Introduction

The unattended call log contains data on unattended call activity. This table holds records for every call made on phone sets associated with a currently unattended machine. Unattended call activity is monitored on machines that are not logged into to the Symposium Agent Server. Unattended call log records are inserted into this table when a call goes to the idle state.

Table

UnattendedCallLog

Field descriptions

UnattendedCallLogID

Description: Unique table id
Type: numeric
Length: 10

CallerNumber

Description: Caller Number
Type: varchar
Length: 20

CalledNumber

Description: Called Number
Type: varchar
Length: 20

Extension

Description: Extension on which call is presented
Type: varchar
Length: 20

ConnectedSeconds

Description: Total number of seconds call was in the connected state
Type: int
Length: 4

HoldSeconds

Description: Total number of seconds call was in the hold state
Type: int
Length: 4

Other Seconds

Description: Total number of seconds call was in states other than connected and hold
Type: int
Length: 4

StartTime

Description: Time call was answered
Type: datetime
Length: 8

Linename

Description: Line device name on which the call arrived
Type: varchar
Length: 20

ExtensionTypeID

Description: Identifier representing the type of extension
Type: int
Length: 4

DirectionID

Description: Identifier representing the direction of the call). The following table lists the valid direction IDs and their meaning:

0	Unknown
1	Inbound
2	Outbound
3	Conferenced

Type: int
Length: 4

CallID

Description: Unique Identifier for the call
Type: int
Length: 4

IVRData

Description: Call data that is attached to the call. This data can originate from an IVR unit or be attached by an application.
Type: int
Length: 4

LastCallStateID

Description: The last call state before the call was release.
Type: int
Length: 4

UserID

Description: Identifier of User associated with the call
Type: numeric
Length: 10

User Session

Introduction

The User Session table contains information specific to a Symposium Agent User Session. Accumulated times for call activity and ACD activity are held in this table, such as the total time in the Connected call state.

Table

UserSession

Field descriptions

SessionID

Description: Unique table ID
Type: numeric
Length: 10

SignOnTime

Description: The session start time; the time the user logged into the Symposium Agent Server
Type: datetime
Length: 8

SignOffTime

Description: The session end time; the time the user logged off the Symposium Agent Server
Type: datetime
Length: 8

WatchDogTime

Description: The last time the client notified the server that it was still up and running.
Type: datetime
Length: 8

UserID

Description: The identifier for the user associated with this session
Type: numeric
Length: 10

MachineID

Description: The identifier for the machine associated with this session
Type: int
Length: 4

CallSummarySnapshotTime

Description: Time that the session record was last updated with call statistics
Type: datetime
Length: 8

InboundIncallsConnected

Description: The total time for all inbound ACD calls that were in the connected state.
Type: int
Length: 4

OutboundIncallsConnected

Description: The total time for all outbound calls on the ACD extension that were in the connected state. Is only valid for those phone sets that allow for outbound calls on an ACD extension.
Type: int
Length: 4

InboundIncallsHold

Description: The total time for all inbound ACD calls that were in the hold state
Type: int
Length: 4

OutboundIncallsHold

Description: The total time for all outbound calls on the ACD extension that were in the hold state. Is only valid for those phone sets that allow for outbound calls on an ACD extension.
Type: int
Length: 4

InboundIncallsOther

Description: The total time for all inbound ACD calls that were in states other than connected and hold state
Type: int
Length: 4

OutboundIncallsOther

Description: The total time for all outbound calls on the ACD extension that were in the states other than connected and hold. Is only valid for those phone sets that allow for outbound calls on an ACD extension.
Type: int
Length: 4

InboundIDNConnected

Description: The total connected time for all inbound calls on the personal DN.
Type: int
Length: 4

OutboundIDNConnected

Description: The total connected time for all outbound calls on the personal DN.
Type: int
Length: 4

InboundIDNHold

Description: The total hold time for all inbound call on the personal DN.
Type: int
Length: 4

OutboundIDNHold

Description: The total hold time for all outbound calls on the personal DN.
Type: int
Length: 4

InboundIDNOther

Description: Total time in call states other than Hold or Connected for all Inbound calls on the personal DN.
Type: int
Length: 4

OutboundIDNOther

Description: Total time in call states other than Hold or Connected for all outbound calls on the personal DN.
Type: int
Length: 4

ActivitySummarySnapshotTime

Description: The last time a record has been updated in the UserSession table.
Type: datetime
Length: 8

ReadyTime

Description: The total time an agent has been in the ready state.
Type: int
Length: 4

NotReadyTime

Description: The total time an agent has been in the not ready state.
Type: int
Length: 4

WalkawayTime

Description: The total time an agent has been in the walkaway state.
Type: int
Length: 4

InboundInCallsCount

Description: The total number of inbound calls on the ACD extension.
Type: int
Length: 4

OutboundInCallsCount

Description: The total number of outbound calls on the ACD extension. Valid only for phone sets that allow for outbound calls on an ACD extension.
Type: int
Length: 4

InboundIDNCount

Description: The total number of inbound calls on the personal DN.
Type: int
Length: 4

OutboundIDNCount

Description: The total number of outbound calls on the personal DN.
Type: int
Length: 4

TerminatedByWatchDog

Description: Flag indicating whether a user session has been ended by the watchdog service due to connectivity problems.
Type: smallint
Length: 2

Activity Log

Introduction

The Activity Log table holds data for ACD activity. This data is stored based on user sessions. The activity ID can be translated to a more user friendly string by looking up the ID in the ACDActivity table.

Table

ActivityLog

Field descriptions

ActivityLogID

Description: Unique table id
Type: numeric
Length: 10

ActivityTime

Description: Date and Time ACD activity took place
Type: datetime
Length: 8

SessionID

Description: unique id for session associated with ACD Activity
Type: numeric
Length: 10

ACDActivityID

Description: Unique Activity id to determine ACD Activity
Type: numeric
Length: 10

LineName

Description: Line device name on which the activity took place
Type: varchar
Length: 50
Length: 2

Call Log

Introduction

The Call Log table holds a records for each call . This data is stored based on user sessions. Call log records are added to the database for each session when the call goes to the idle state.

Table

CallLog

Field descriptions

CallLogID

Description: Unique table id
Type: numeric
Length: 10

CallerNumber

Description: Caller Number
Type: varchar
Length: 20

CalledNumber

Description: Called Number
Type: varchar
Length: 20

Extension

Description: Extention on which call is presented
Type: varchar
Length: 20

ConnectedSeconds

Description: Total number of seconds call was in the connected state
Type: int
Length: 4

HoldSeconds

Description: Total number of seconds call was in the hold state
Type: int
Length: 4

Other Seconds

Description: Total number of seconds call was in states other than connected and hold
Type: int
Length: 4

StartTime

Description: Time call was answered
Type: datetime
Length: 8

Linename

Description: Line device name on which the call arrived
Type: varchar
Length: 20

ExtensionTypeID

Description: Identifier representing the type of extension
Type: int
Length: 4

DirectionID

Description: Identifier representing the direction of the call). The following table lists the valid direction IDs and their meaning:

0	Unknown
1	Inbound
2	Outbound
3	Conferenced

Type: int
Length: 4

CallID

Description: Unique Identifier for the call
Type: int
Length: 4

IVRData

Description: Call data that is attached to the call. This data can originate from an IVR unit or be attached by an application.
Type: int
Length: 4

LastCallStateID

Description: The last call state before the call was release.
Type: int
Length: 4

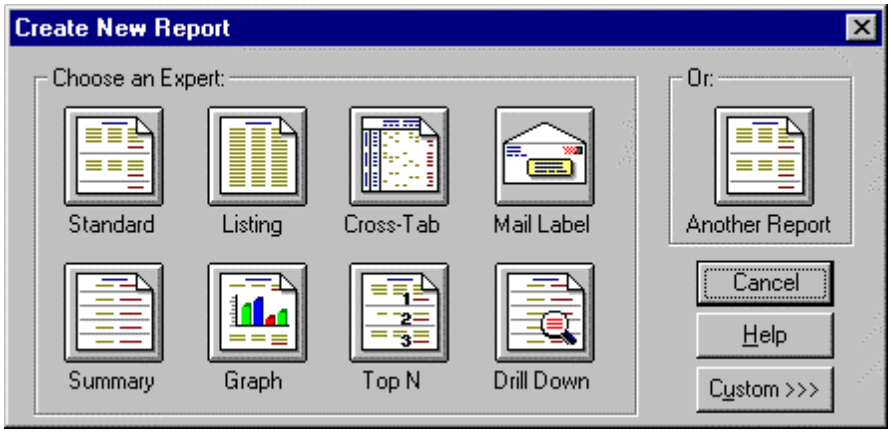
SessionID

Description: unique id for session associated with Call
Type: numeric
Length: 10

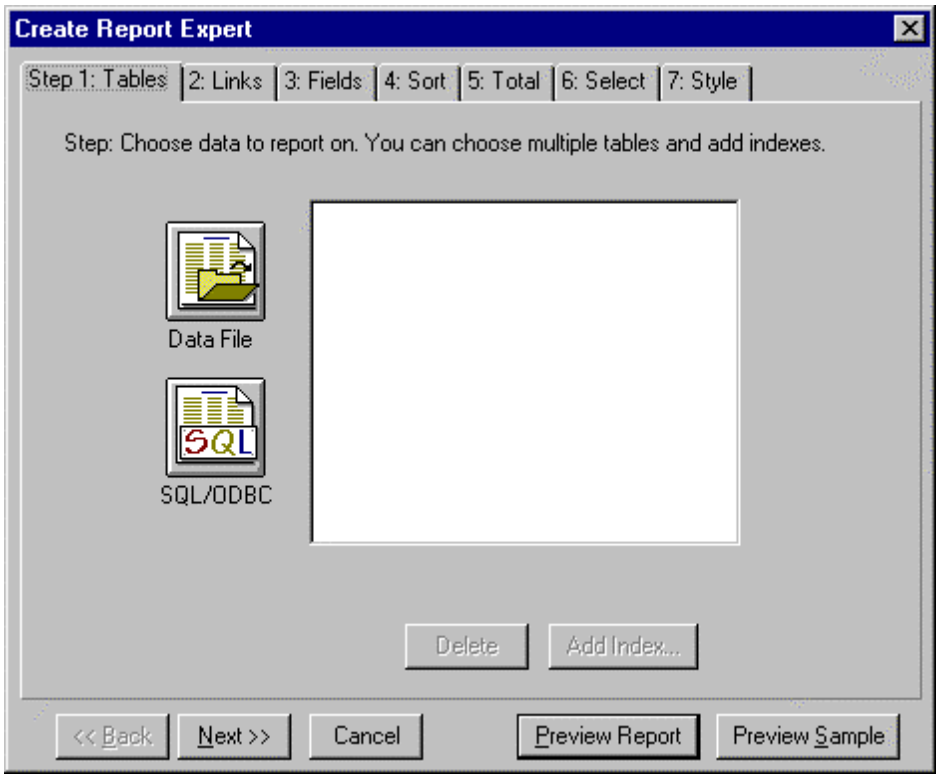
Sample Report using Crystal Reports

This sample assumes you have training in how to use Crystal Reports. This example uses Crystal Reports for Visual Basic 5.0, which uses a subset of the features that are available through the full Crystal Reports package. This sample will demonstrate how to create a simple call log report.

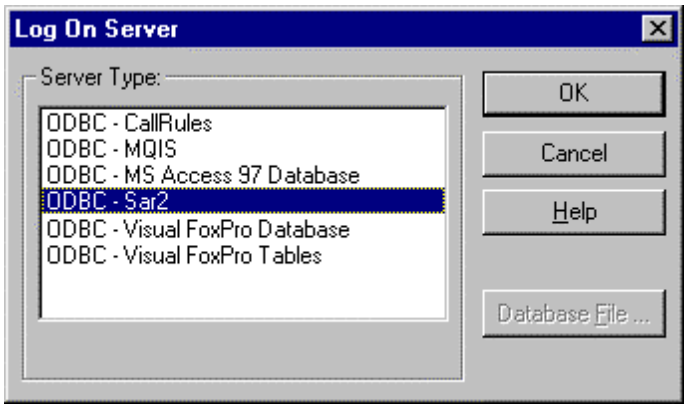
1 Click on the “Create New Report” icon on the main window toolbar. From the “Create New Report” dialog select the desired report type (example uses Standard).



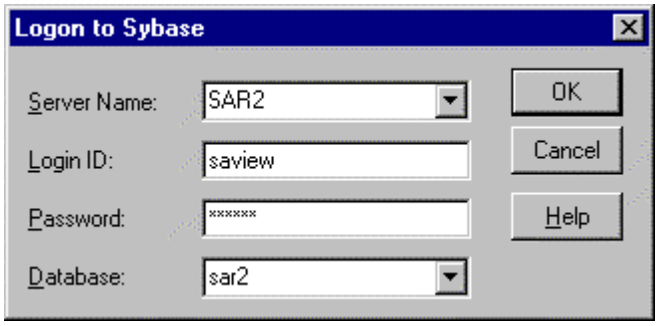
3 Select the type of data to use for the report (example uses SQL/ODBC).



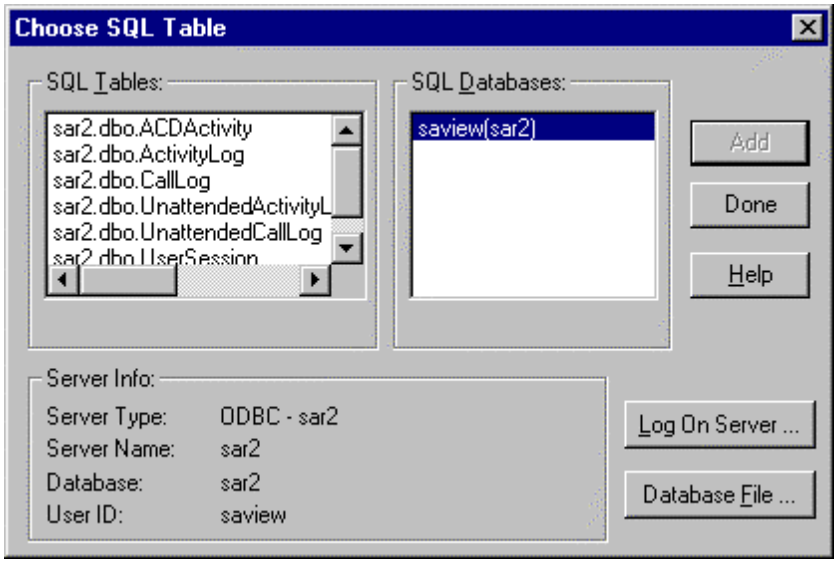
The “Log on Server” dialog appears.



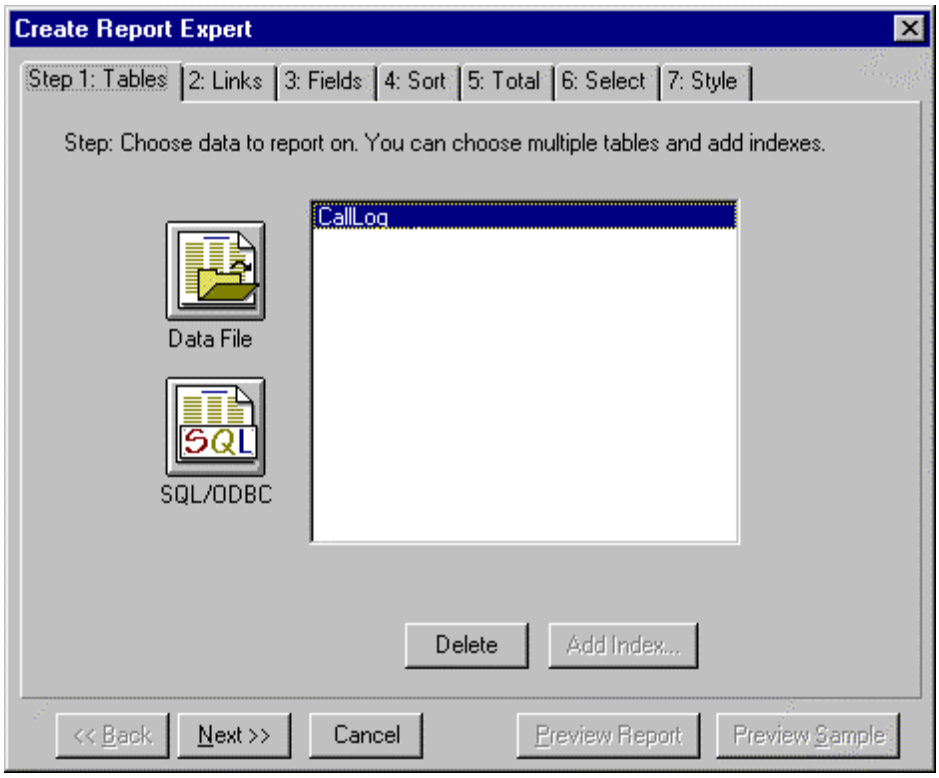
4 From the “Log on Server” dialog, select the server name (example uses SAR2). Log on to Sybase with the saview user and password.



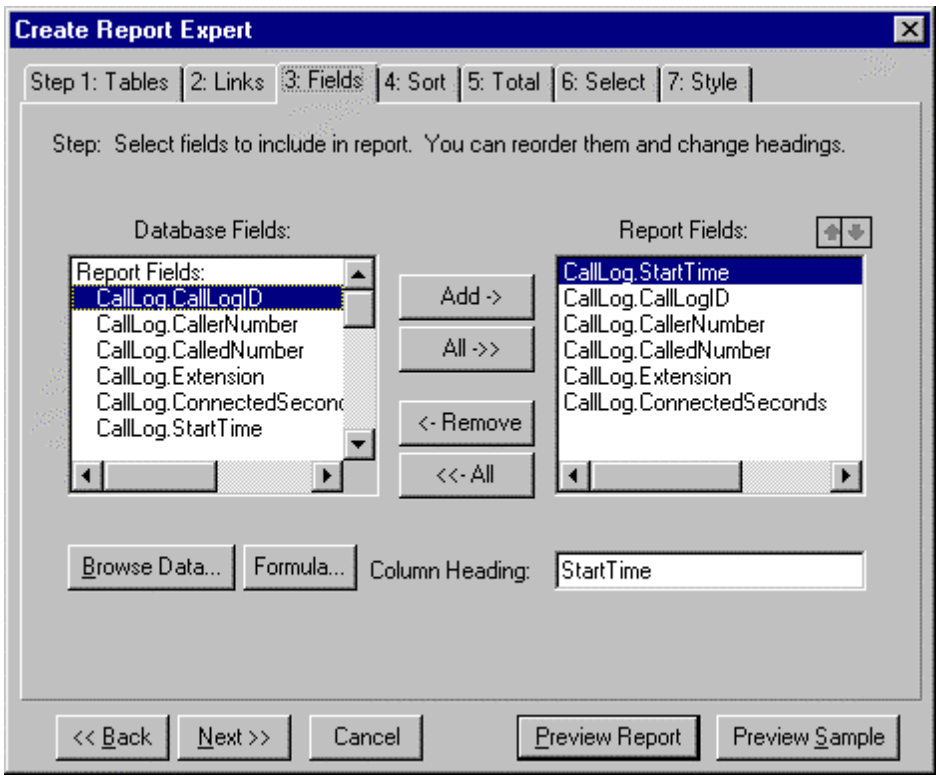
5 The “Choose SQL Table” dialog appears. Select the desired tables.



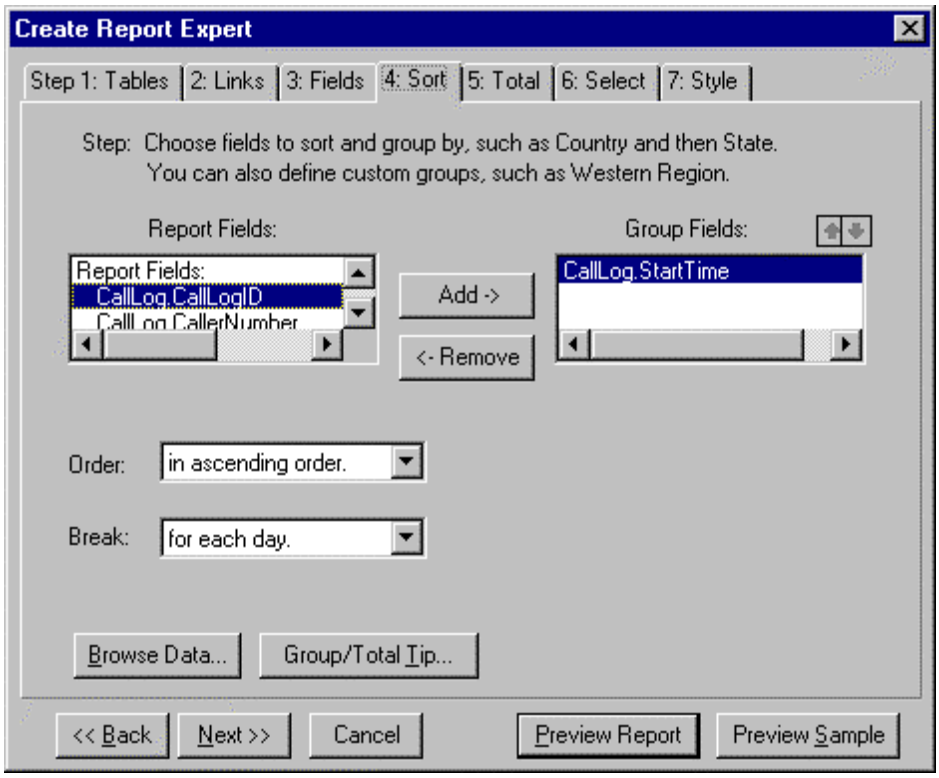
6 The tables selected will appear in the listbox for the Tables tab.



7 From the Fields Tab, select the Fields to display in the report.

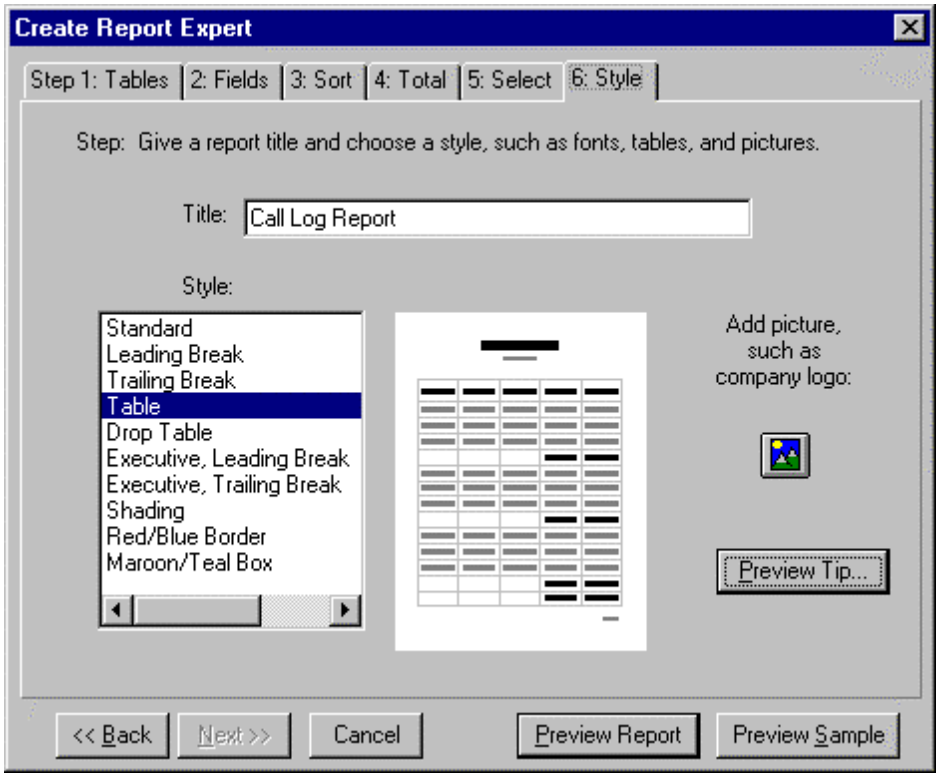


8 From the Sort Tab, select the Field(s) by which to sort data



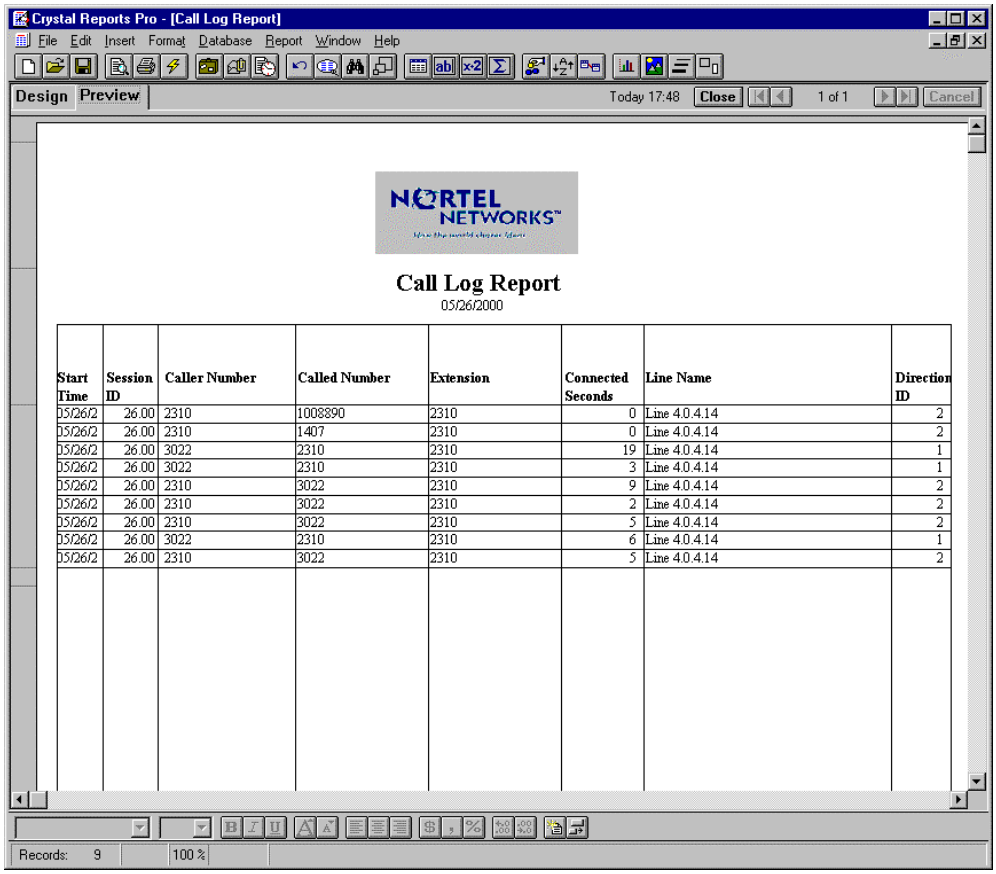
9 From the Style tab select the report format.

This example uses the Table format. You can also specify an image to include in the report. The other tabs are optional.



10 Click Review Report to view the report.

The sample report results:



Chapter 5

Outbound dialing using Symposium Agent

In this chapter

Dial features 76

Dial feature

Symposium Agent provides an interface for third party applications to make outbound calls. This component is called NNCO.dll; it allows third party applications to dial a number. This file is located in the NNLogin folder on each client.

Adding outbound dialing to your application requires that the SA client be installed on the system on which your application is being developed.

NNCO provides the following public interface:

Properties:
- LineName
- LangID

Methods:
- Dial

NNCO has the PROGID of NNCO.cCallObject.

NNCO automatically opens the line specified in the LineName property. By default it will get this line name from the Symposium Agent database through the machine/line association.

The LangID property is the locale ID for the language to use to display error messages. By default LangID is set to English (1033).

The Symposium Agent client must be running in order for NNCO to initialize. If the client is not running, NNCO will display "Failed to initialize Dialer" error message.

Here are two examples on how to use NNCO to dial a number. The first one is a Visual Basic 6.0 code sample. The second example shows how to use NNCO from a web page.

VB Example:

This example is from a VB form that has three text boxes. The text boxes ask for the number to dial, the line name for the line to dial from, and the language ID for the language in which to display error messages.

```
Dim nnco As New nnco.cCallObject

Private Sub cmdDial_Click()
   nnco.Dial Text1.Text
End Sub

Private Sub cmdInitialize_Click()

   nnco.LineName = Text2.Text
   If Len(Text3.Text) <> 0 Then
      nnco.LangID = Text3.Text
   End If
```

```
    If nnco.Initialize = False Then
       MsgBox "Failed to initialize TAPI"
    End If

End Sub
```

HTML example:

This sample web page has one text box for the number to be dialed and one button. When the user presses the button, the NNCO object is intialized and the number is dialed.

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>NNCOTest</TITLE>
<SCRIPT LANGUAGE="VBSCRIPT"><!--

sub ShowFormatedDateTime()
        NNCO.Dial text1.value
end sub

sub InitializeNNCO()
        NNCO.Initialize
end sub

//--></SCRIPT>
</HEAD>
<BODY onload="InitializeNNCO();">
<OBJECT ID="NNCO" CLASSID="CLSID:7B52F8AC-BC8E-11D4-88FD-0010A4E508D5">
</OBJECT>

<P> </P>
<INPUT type="text" id=text1 name=text1>
<INPUT type="button" value="Button" id=button1 name=button1 onclick="Dail()">
</BODY>
</HTML>
```

Chapter 6

Active Server Page examples

In this chapter

Introduction	79
An Active Server Page sample (NNWEB.ASP)	80
Sample source for switchboard.asp	82
Sample source for flowers.asp	83
Sample source for pizza.asp	84
Sample source for unknown.asp	85

Introduction

This chapter contains sample Active Server Pages (ASP) code. The sample code displays the call state and telephony information passed from agentCompass to the agentTabs. To use the ASP files, the web server must be Microsoft Internet Information Server 3.0 or higher. You can copy the following sample and then publish it to a web directory that has Execute permissions. These ASP files include sample code for using the call information associated with call states Offering, Connected, and Idle. They also include a simple switchboard application that redirects the browser to a different URL based on the called number.

An Active Server Page sample (NNWEB.ASP)

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-
1252">
<TITLE>Symposium Agent Sample Web Application</TITLE>
</HEAD>
<BODY LINK="#0000ff" VLINK="#800080">
<P><!doctype HTML public "-//W3O//DTD W3 HTML 3.0//EN"></P>
<P><HR></P>

<H1 ALIGN="CENTER">Symposium Agent Sample Web Application </H1>
<P><HR></P>

<P ALIGN="CENTER">
<script LANGUAGE="JavaScript">
<!--
// This line executes when the script tag is parsed.

Now = new Date();
if (Now.getHours() < 12)
  document.write(" Good morning!  ");
else if (Now.getHours() < 17)
  document.write(" Good afternoon!  ");
else
  document.write(" Good evening!  ");

document.write("Welcome to the world of Symposium Agent, it's ");
document.write(" " + Now.toLocaleString());
//-->
</script>
</P>

<BR> <BR>
<div align="center"><center>

<table border="1">
<TR><TD VALIGN="MIDDLE" COLSPAN=2>
<P ALIGN="CENTER"><B>Telephony Information</B></TD>
</TR>
   <tr>
      <td align="center">Telephony Parameters</td>
      <td align="center">Current Value</td>
   </tr>
   <tr>
      <td>Call State</td>
      <td><% = Request("State") %>   </td>
   </tr>
   <tr>
      <td>Caller Number</td>
      <td><% = Request("CallerNumber") %>   </td>
   </tr>
   <tr>
      <td>Caller Name</td>
```

```
      <td><% = Request("CallerName") %>   </td>
    </tr>
    <tr>
      <td>Called Number</td>
      <td><% = Request("CalledNumber") %>   </td>
    </tr>
    <tr>
      <td>Called Name</td>
      <td><% = Request("CalledName") %>   </td>
    </tr>
    <tr>
      <td>Call Data</td>
      <td><% = Request("CallData") %>  </td>
    </tr>
    <tr>
      <td>Call ID</td>
      <td><% = Request("CallID") %>   </td>
    </tr>
    <tr>
      <td>Extension</td>
      <td><% = Request("Extension") %>   </td>
    </tr>
    <tr>
      <td>Line Name</td>
      <td><% = Request("LineName") %>   </td>
    </tr>
    <tr>
      <td>Login ID</td>
      <td><% = Request("LoginID") %>   </td>
    </tr>
    <tr>
      <td>Language ID</td>
      <td><% = Request("LangID") %>   </td>
    </tr>
    <tr>
      <td>Trunk Info</td>
      <td><% = Request("TrunkInfo") %>   </td>
    </tr>
    <tr>
      <td>Time Zone</td>
      <td><% = Request("TimeZone") %>   </td>
    </tr>
    <tr>
      <td>Version</td>
      <td><% = Request("Version") %>   </td>
    </tr>
    <tr>
      <td>ACD Agent ID</td>
      <td><% = Request("ACDAgentID") %>   </td>
    </tr>

  </table>
  </center></div><div align="center"><center>

  <P><HR></P>
  </BODY>
  </HTML>
```

Sample source for switchboard.asp

This is an Active Server Page (ASP) that looks at the CalledNumber to redirect the browser to a new URL.

```
<%
' $Date: 03/26/99 11:40a $
' $ModTime: $
' $Revision: 2 $
' $Workfile: switchboard.asp $
' $Copyright (C) 1998-1999, Nortel Networks. All rights reserved. $
Dim sCalledNumber
Dim sRedirectURL
sCalledNumber = Request("CalledNumber")
Session("CallerNumber") = Request("CallerNumber")
Session("CalledNumber") = sCalledNumber
Session("CallData") = Request("CallData")
Session("CallID") = Request("CallID")
Session("State") = Request("State")
Select Case sCalledNumber
Case "1411"
sRedirectURL = "Flowers.asp"
Case "1412"
sRedirectURL = "Pizza.asp"
Case Else
sRedirectURL = "Unknown.asp"
End Select
'Redirect requires IE4 clients to have HTTP 1.1 checked under the Advanced
settings.
'Response.Redirect sRedirectURL
%>
<% response.addheader "pragma","no-cache" %>
<% response.expires=0 %>
<% response.expiresabsolute=#Feb 1,1999 00:00:00# %>
<% response.cachecontrol="public" %>
<meta HTTP-EQUIV="Refresh" CONTENT="0; URL=<%=sRedirectURL%>">
<HTML>
Loading page, please wait...
</HTML>
```

Sample source for flowers.asp

This is an Active Server Page (ASP) that looks at the call state, State.

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<TITLE>Welcome to the online flower store</TITLE>
<BODY>
Thanks for calling for flowers<BR>
Telephony information<BR>
<TABLE>
<TR>
<TD>Call state</TD>
<TD> <% = Session("State") %> </TD>
</TR>
<TR>
<TD>CallerName</TD>
<TD> <% = Session("CallerName") %> </TD>
</TR>
<TR>
<TD>CallerNumber</TD>
<TD> <% = Session("CallerNumber") %> </TD>
</TR>
<TR>
<TD>CalledName</TD>
<TD> <% = Session("CalledName") %> </TD>
</TR>
<TR>
<TD>CalledNumber</TD>
<TD> <% = Session("CalledNumber") %> </TD>
</TR>
<TR>
<TD>CallData</TD>
<TD> <% = Session("CallData") %> </TD>
</TR>
<TR>
<TD>CallID</TD>
<TD> <% = Session("CallID") %> </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Sample source for pizza.asp

This is an Active Server Page (ASP) that looks at the call state, State.

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<TITLE>Welcome to the online Pizza store</TITLE>
<BODY>
Thanks for calling for pizza<BR>
Telephony information<BR>
<TABLE>
<TR>
<TD>Call state</TD>
<TD> <% = Session("State") %> </TD>
</TR>
<TR>
<TD>CallerName</TD>
<TD> <% = Session("CallerName") %> </TD>
</TR>
<TR>
<TD>CallerNumber</TD>
<TD> <% = Session("CallerNumber") %> </TD>
</TR>
<TR>
<TD>CalledName</TD>
<TD> <% = Session("CalledName") %> </TD>
</TR>
<TR>
<TD>CalledNumber</TD>
<TD> <% = Session("CalledNumber") %> </TD>
</TR>
<TR>
<TD>CallData</TD>
<TD> <% = Session("CallData") %> </TD>
</TR>
<TR>
<TD>CallID</TD>
<TD> <% = Session("CallID") %> </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Sample source for unknown.asp

This is an Active Server Page (ASP) that looks at the call state, State.

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<TITLE>Thank you for calling (unknown number)</TITLE>
<BODY>
Telephony information<BR>
<TABLE>
<TR>
<TD>Call state</TD>
<TD> <% = Session("State") %> </TD>
</TR>
<TR>
<TD>CallerName</TD>
<TD> <% = Session("CallerName") %> </TD>
</TR>
<TR>
<TD>CallerNumber</TD>
<TD> <% = Session("CallerNumber") %> </TD>
</TR>
<TR>
<TD>CalledName</TD>
<TD> <% = Session("CalledName") %> </TD>
</TR>
<TR>
<TD>CalledNumber</TD>
<TD> <% = Session("CalledNumber") %> </TD>
</TR>
<TR>
<TD>CallData</TD>
<TD> <% = Session("CallData") %> </TD>
</TR>
<TR>
<TD>CallID</TD>
<TD> <% = Session("CallID") %> </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Index

Active Server Pages, 3, 76
agentBrowser, 34, 35, 38, 39
agentCompass, 3, 40, 76
AgentCompass, 3, 40
agentTab
  Telephony enabled, 42
agentTabs, 3, 7, 34, 36, 38, 44
ASP, 3, 37, 76, 77
call rules, 3, 6, 30, 34, 38, 39
Call Rules Runtime, 6, 34, 38
Crystal Reports, 70
Database dictionary, 57
Dialog box
  New Application, 15, 16, 20, 21
  Rule Wizard – Finished, 30
  Rule Wizard – Step 1, 28
  Rule Wizard – Step 2, 28
  Trigger Wizard – Finished!, 17, 22
  Trigger Wizard – Step 1, 10, 15, 20, 39
  Trigger Wizard – Step 2, 21
  Trigger Wizard – Step 3, 16, 21
  Verify Classname, 21
Dialog Box
  Rule Wizard – Step 3, 29
Entity Relationship Diagram, 55
IIS, 37
Internet Explorer 4.0, 5
NNLogin, 34
ODBC System DSN, 52
OLE, 3, 6, 16, 21, 34, 38, 40
Open Database Connectivity (ODBC), 46
POST method, 34, 35
Service Pack 4, 5
Sybase, 47
Sybase server, 48
Symposium Agent 2.3 Server, 5
Symposium Agent Call Rules Explorer, 3
Symposium Agent Release 2.3, 3
TAPI, 6, 34
The Rule Wizard – Step 5, 30
Trigger Wizard, 10, 15, 20
URL, 3, 34, 35, 38, 40, 42, 43, 76
user profile, 43
web server, 3, 34, 35, 37, 43, 44, 76
Windows NT 4.0, 5

Symposium Agent
Programmer's Guide

Nortel Networks
9701 Data Park
Minnetonka, MN. 55343
USA

Copyright © 2001 Nortel Networks, All Rights Reserved.

Product release: 2.3
Document release: Standard 2.0
Date: January 2001

NORTEL NETWORKS™

How the world shares ideas.